Spring 5-2022

Data and Algorithmic Modeling Approaches to Count Data

Andraya Hack

Murray State University Honors College

HONORS THESIS

Certificate of Approval

Data and Algorithmic Modeling Approaches for Count Data

Andraya Hack
May 2022

Approved to fulfill the
requirements of HON 437

_____
Dr. Christopher Mecklin, Professor
Department of Mathematics and Statistics

Approved to fulfill the
Honors Thesis requirement
of the Murray State Honors
Diploma

_____
Dr. Warren Edminster, Executive Director
Honors College

Examination Approval Page

Author: Andraya Hack

Project Title: Data and Algorithmic Modeling Approaches for Count Data

Department: Math and Statistics

Date of Defense: April 25th, 2022

Approval by Examining Committee:

_____ _____

(Dr. Christopher Mecklin, Advisor) (Date)

_____ _____

(Dr. Gopal Nath, Committee Member) (Date)

_____ _____

(Dr. Manoj Pathak, Committee Member) (Date)

Data and Algorithmic Modeling Approaches for Count Data

Submitted in partial fulfillment of the requirements for

The Murray State University Honors Diploma

Andraya Hack

May 2022

ABSTRACT

Various techniques are used to create predictions based on count data. This type of data takes the form of a non-negative integers such as the number of claims an insurance policy holder may make. These predictions can allow people to prepare for likely outcomes. Thus, it is important to know how accurate the predictions are. Traditional statistical approaches for predicting count data include Poisson regression as well as negative binomial regression. Both methods also have a zero-inflated version that can be used when the data has an overabundance of zeros. Another procedure is to use computer algorithms, also known as machine learning, for predictions. Two specific algorithms used here are artificial neural networks (ANN) and k-nearest neighbors (KNN). This project aims to consider both traditional statistical modeling and algorithmic modeling to find which technique is the most accurate and therefore most effective. This will be accessed by using two datasets to test the assorted models.

KEYWORDS: Prediction, Count Data, Poisson Regression, Zero-Inflated, ANN, KNN, algorithms

Table of Contents

1. INTRODUCTION

A useful field of study is the art of prediction using sample data. There are a multitude of events or outcomes that can be forecasted. Many people are familiar with weather predictions. These forecasts were produced using the analysis of past data to create a tool to anticipate future events. Count data is a type of data that can be incredibly useful to predict. Count data encompasses any information that can be counted as the name suggests. That means the data will be in a non-negative integer format and represents the quantity of the variable under consideration.

One area where count data is important is in the actuarial field. Actuaries assess uncertainties as well as the likelihood of losses, especially in the insurance field. To establish and monitor risk actuaries may want to predict how many insurance claims would be made in a particular period. In addition, they may want to predict how many car accidents there will be or how many doctor visits a policy holder will have. These are examples of count predictions that may affect the cost of insurance policies and the risk a company takes on when insuring a person.

The biology discipline will also commonly use count data and biologists desire predictions for such information. Examples of this may be how many members of a specific species is found in an area. This includes animals or plants in an environment or could be the number of colonies of bacteria in a given location. Other examples could be to predict the number of people who will contract a disease or the quantity of species or organisms that will be born or die during a chosen time span.

There are a multitude of methods to predict count data. One approach is traditional statistical modeling. These models can aid in actually demonstrating what is happening via statistical

inference. This means that the model can help explain the underlying mechanisms at work. This can be helpful when presenting this information to others. Another approach is to use computer algorithms, often called machine learning, to produce the predictions. One component that may turn researchers away from algorithms is that they often do not identify the underlying processes. The input is taken and then an assortment of transformations are made by the program to produce the output. The predictions are the outputs. The uncertainty about the process that the inputs undergo to create the results does not necessarily tell those studying the algorithm anything about the real-world processes that are occurring.

The models used in a classical or traditional approach to predict counts utilize the Poisson distribution and the negative binomial distribution. The Poisson model uses a parameter that is the rate of occurrence per interval to determine the number of events in a given time period. Meanwhile the negative binomial uses a parameter which is the probability of an event to determine a count of trials that are necessary. However, the data for some scenarios may be skewed towards zero. To fix the lack of fit caused by an overabundance of zeros known as overdispersion, one can use a zero-inflated version of either model.

There are numerous algorithms that can be used to predict count data. One algorithm is called Artificial Neural Network or ANN. This algorithm is supposed to mimic pathways of neurons in the brain. ANN relies on inputs that are used to compute hidden layers before returning an output. Another algorithm is the k-nearest neighbor or KNN which uses boundaries to classify the data allowing the number of an event to be predicted. Other algorithms that could be used are decision trees like the classification and regression tree also known as CART, or a tree augmented naïve Bayes model also known as TAN.

No matter what tool is used to make predictions there are crucial elements of the results to consider. The most significant component is the accuracy of the method used. The accuracy is a measure of how correct the prediction is. The various techniques to anticipate counts each will have different accuracies. The higher the accuracy the better the method is at making predictions.

The goal of this project is to compare traditional models with algorithms and their ability to predict count data. To assess which methods are better, the accuracy will be measured. To do this two data sets will be used to test Poisson linear regression, zero-inflated Poisson linear regression, ANN, and KNN. From these tools the accuracies will be measured and then compared. One data set will have an actuarial focus with information about car insurance claims. The other data set will have a concentration on how many articles biochemist students publish. The hypothesis is that the algorithms with have better accuracy compared with the traditional models.

2. LITERATURE REVIEW

2.1 Count Data and Linear Regression

Data comes in many forms and can hold a vast amount of information. However, just looking at the data via summary statistics will not necessarily reveal that information, especially when working with multiple variables and large datasets. Various approaches can be made to examine the data and gain insights. The conclusions that come from analysis involve how the variables are correlated to each other. Predictions are a product of such insights. One approach for prediction is using generalized linear models. These models will use nonlinear data to synthesize a linear approximation. The most basic statistical method is the linear regression model. When there is a single predictor variable the simple linear regression is used the approximation comes from:

$$Y = \beta_0 + \beta_1 X + \epsilon \qquad \text{(James et al. 2021, p. 61)}$$

Where β_0 and β_1 are coefficients that are to be estimated using the data, while Y is what is to be predicted and X is the variable used to make the prediction. The ϵ term stands for the error that is unexplained by the model. However, it is unlikely for there to be only one variable that affects the prediction. Thus, multiple linear regression, which allows for more variables, is more commonly use with an equation as follows:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta p Xp + \epsilon \qquad \text{(James et al. 2021, p. 72)}$$

This model uses p variables to predict Y. In either case the coefficients also known as parameters are chosen based on which values minimize the sum of squared residuals given by:

$$\text{RSS} = \sum_{i=0}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - \cdots - \hat{\beta}_p x_{ip})^2 \qquad \text{(James et al. 2021, p. 72)}$$

To find these parameter estimates some sort of programming environment such as R or R Studio would be used. When performing multiple linear regression in R the response variable (also known as y) is assumed to be "independent and normally distributed with a constant variance" (Bolker 2008, p. 300). Something important to note is that the error associated with this model is based on the RSS displayed earlier. The residual standard error (RSE), which is also known as the square root of the MSE, uses the following equation:

$$RSE = \sqrt{\frac{RSS}{n-p-1}}$$ (James et al. 2021, p. 80)

In the context of this project, the data will involve counts. Counts can take the form of zero in addition to any positive integer. This means the data will not fit a normal distribution which is an assumption of linear regression. Count data is discrete while the normal distribution is continuous, which presents a problem. With large enough counts, an approximation using a normal distribution can be used due to the Central Limit Theorem (CLT) (Hogg et al. 2015, p.122). However, with low count values, this may result in large error and an unreliable prediction. Therefore, it may be better to use a generalized linear model that uses a non-normal family (Bolker 2008, p. 308). Such a model could possibly improve the analysis.

2.2 Poisson and Zero-Inflated Poisson Distributions

One method for modeling count data that is commonly used is Poisson regression. The Poisson distribution can be used to approximate the number or rate of events that occur during a specific period of time, space, or unit (Hogg et al. 2015, p. 79). In order for the Poisson distribution to be used it is assumed that the count behaves as a Poisson process. There are three conditions for whether the occurrences follow a Poisson process. The first is that in separate intervals the counts are independent of each other. This means that the number of events in one period has no effect on the events occurring in a different period. The second condition is that in a sufficiently small section of the given interval of length d, the probability of one occurrence is d times the parameter λ. The final condition is that in a sufficiently small subinterval, the probability of having two or more events is zero (Hogg et al. 2015, p. 79). When these requirements are met, the counts work as a Poisson process allowing the Poisson distribution to be used.

The most prevalent version of the Poisson distribution contains a single parameter λ (Bolker 2008, p. 122). The parameter λ represents the "average density or arrival rate" (Bolker 2008 p. 122). Technically speaking the Poisson distribution could produce counts to infinity, though practically such large counts will essentially have a probability of zero. Thus, the Poisson is helpful when there is not a limit on the counts. Another fact about the Poisson distribution is that the variance is equal to its mean which are both equal to λ (Bolker 2008, p. 122). The Poisson distribution is as follows:

$$f(x) = \frac{\lambda^x e^{-\lambda}}{x!}, \quad x=0, 1, 2, \ldots, \lambda > 0 \quad \text{(Hogg et al. 2015, p. 80)}$$

Where x is the number of counts, and the parameter λ is the mean occurrences per interval. This distribution can then be the basis of a type of generalized linear model called Poisson regression. To create a Poisson regression the log is usually used as the link function. All of the generalized linear models "are fit by a process called iteratively reweighted least squares, which overcomes the basic problem that transforming the data to make them linear also changes the variance" (Bolker 2008, p. 309).

One issue with the Poisson distribution is that there might be more zero counts in the dataset than the distribution would predict. This causes overdispersion of the data. The overdispersion and skew in the count data results in a higher variance than a Poisson distribution would have (Bolker 2008, p. 311). A solution to this problem is to use the zero-inflated Poisson regression which "is used to model count data that has an excess of zero counts" ("Zero-Inflated Poisson Regression" 2021). This model consists of two parts separating the extra zeros from the Poisson distribution. These two components are a Poisson distribution as well as a logit model used to predict the extra zeros ("Zero-Inflated Poisson Regression" 2021). When using X_i for the counts, a probability of p_i, and a mean of λ_i the zero-inflated Poisson model, often called a "mixture distribution", is a mixture of

$$P(X_i = 0) = p_i + (1 - p_i)e^{-\lambda_i}$$

$$P(X_i = k) = \frac{(1-p_i)e^{-\lambda_i}\lambda_i{}^k}{k!} \qquad k = 1,2,\ldots \qquad \text{(Loeys et al. 2012)}$$

These two components are designed to accommodate an abundance of zero counts, the goal being to get the most accurate predictions.

2.3 Negative Binomial and Zero-inflated Negative Binomial Distributions

Another model that can be used for counts is the negative binomial distribution. This distribution relies on a fixed number of successes, and it looks at the number of trials it takes to reach the fixed number of successes (Hogg et al. 2015, p. 74). This can also be viewed as counting the number of failures. The main benefit to using the negative binomial distribution is that while it is discrete "like the Poisson distribution, but its variance can be larger than its mean" (Bolker 2008, p. 124). This allows the data to be overdispersed meaning the distribution can be more skewed or clustered. The equation for the negative binomial distribution is

$$f(x) = \binom{x-1}{r-1} p^r (1-p)^{x-r} \qquad \text{x=r, r+1, r+2, …} \qquad \text{(Hogg et al. 2015, p. 74)}$$

In this equation x is the number of trials, r is the number of successes, and p is the probability of a success.

An interesting property of the negative distribution is it can be expressed using Gamma functions with k as the shape parameter and μ as the mean parameter. This distribution will have a "mean μ and overdispersion parameter k" (Bolker 2008, p. 124). This results in the following equation:

$$f(x) = \frac{\Gamma(k+x)}{\Gamma(k)x!}\left(\frac{k}{k+\mu}\right)^k \left(\frac{\mu}{k+\mu}\right)^x \qquad \text{(Bolker 2008, p. 125)}$$

This is called the "ecological parameterization" (Bolker 2008, p. 124). The primary function of this format of the negative binomial distribution is that it related to the Poisson distribution and uses an overdispersion parameter.

The negative binomial distribution is the basis of another regression that belongs to the generalized linear models. Thus, like the Poisson regression it is fitted by "iteratively reweighted

least squares" (Bolker 2008, p. 309). Unlike the Poisson regression the negative binomial regression is based on the logit link in R (Bolker 2008, p. 308). The logit function takes the log of the odds ratio. The negative binomial distribution can still have an overabundance of zeros despite the ability to have a larger variance. To remedy this issue a zero-inflated negative binomial distribution can be used. This is done by "iteratively fitting the k (overdispersion) parameter" (Bolker 2008, p. 312). Similar to the zero-inflated Poisson distribution the zero-inflated negative binomial distribution can get more accurate results when there are too many zeros in the count data.

2.4 ANN and KNN Algorithms

In contrast to the models that use a data modeling approach which is the traditional method in statistics, there is the machine learning approaches which use algorithmic models. These are newer methods that have developed greatly as computing power has increased. Machine learning is also referred to as algorithmic modeling. This process takes inputs before it puts them through an intricate and at least partially unknown process to get outputs which would be the predictions (Brieman 2001, p. 205). A branch of machine learning called deep learning uses neural networks for analyzing the data (James et al. 2021, p. 403). Before the data goes through an algorithm it has to be preprocessed. This preprocessing converts the raw data into a form the algorithms can use (Skorikov et al. 2021).

One of the first operations required is to eliminate the “holes”, or missing data, in the data. There is likely information missing from the data collection. These gaps can cause issues so they must be fixed in some way such as removing them. Missing data can also be imputed, in where the missing data can be predicted with a variety of methods. How the missing data is imputed or removed can cause severe bias to be introduced into the model. Another manipulation that may need to be made is to transform categorical data into numbers (Skorikov et al. 2021). This includes items such as gender or location which can be assigned numerical values.

Once such actions are completed the data needs to be divided into usable portions for both training and testing. Machine learning involves training the program so it can predict the targeted information later. A procedure to do this is called k-fold cross-validation. This technique divides the data into k equal partitions. One portion is used as a test while the others are for training the algorithm. Cross- validation is introduced to ensure that the divisions are fairly equal

and that there is not a skew with a minority overrepresented in one set. The goal of this is to lower bias as well as variance. The cross-validation (CV) has the equation

$$CV = \frac{1}{k}\sum_{i=1}^{k} PM_i \qquad \text{(Simsek et al. 2020, p. 71)}$$

The PM refers to "the performance metric employed for evaluating and comparing the model performances" (Simsek et al. 2020, p. 71).

After the data is in a usable form it is put into an algorithm. One type of algorithm is Artificial Neural Networks, also known as ANNs. These ANNs work in a way that is similar to neurons in the brain. "In ANN, the information flow through each neuron occurs in an input-output manner" (Simsek et al. 2020, p. 71). Every ANN has at least three layers; an input layer, a hidden layer, and an output layer ("Implementation of Artificial Neural Network in Python" n.d.). There can be multiple hidden layers. The input layer uses the preprocessed data to send to the first hidden layer. Each hidden layer is given weighted sums of the inputs by an activation function before processing them using the activation function (Simsek et al. 2020, p. 71). During training the weights are adjusted until no more improvement is made. The final layer which is the output can come in a few different forms. There can be a singular output that is either continuous or binary. There can also be multiple outputs with a categorical variable ("Implementation of Artificial Neural Network in Python" n.d.).

Another algorithm to use is K- Nearest Neighbors, also known as KNN. These algorithms are "used to solve the classification model problems" ("k-nearest neighbor algorithm in Python" 2021). KNN utilizes a chosen k which is a distance to create decision barriers. Data points within these decision barriers with be assigned to the specified category. These allow the algorithm to predict whether a point is within that barrier (James et al. 2021 p. 39). The smaller k leads to a

more complex model so a challenge is to select a k value that is accurate but not too complicated ("k-nearest neighbor algorithm in Python" 2021).

No matter the algorithm used their accuracy needs to be tested. Once the algorithm is trained it is used on the test set. This allows those performing the experiment to compare the predictions to actual data. The accuracy is the percentage correct of the prediction (Skorikov et al. 2021). This is often computed by creating predictions with the test portion that was separated out early on. Those predictions are then compared to the actual values associated with the test set. The higher the accuracy is the better predictor that algorithm is judged to be.

3. METHODS

To compare the traditional statistical models to algorithmic models, both types of models will be fit. For this project, the data will consist of counts and will be large enough for the machine learning algorithms to be able to train with them. The datasets used in this project are described in the next chapter.

For the traditional statistical models, the R statistical programming environment is used. There are several packages necessary for the analysis. These packages include *tidyverse* (Wickham et al 2019), *mosaic* (Pruim et al 2017), *MASS* (Venables and Ripley 2002), *countreg* (Zeileis and Kleiber 2020), *vcd* (Meyer 2021), *car* (Fox and Weisberg 2019), and *pscl* (Jackman 2020). The packages are installed and then loaded. Next the dataset must be loaded for this project that required downloading and accessing another R package, the *insuranceData* (Wolny-Dominiak and Trzesiok 2014) package for the car insurance data set and the *biochemist* dataset is located in the already loaded *pscl* package. To get a sense of the data the basic descriptive statistics are compiled, and graphs are drawn. Once familiar with the contents of the dataset, it is necessary to figure out what variable is to be predicted and which variables are to be used to create the prediction.

After setting up the environment and the dataset is prepared the models for prediction can be created. For the Poisson the generalized linear model function, which is called *glm*, is used with the family set to Poisson. The *summary* function displays the coefficients, standard error, and z or t test statistics that the model uses.

Two ways to analyze the effectiveness of this model is to use a function called *Anova* and another is called the goodness of fit test called the *goodfit* function. Anova stands for analysis of variance. For a generalized linear model, this with be based on an analysis of deviance, rather

than sum of squares as seen in linear models. The *Anova* function returns a table with chi-square test statistics and p-values for each variable. The goodness of fit test returns results function for the fit of the entire model. The graph of the goodness of fit test shows a visualization of the observed versus predicted counts in a hanging rootogram. When the bar for zero counts hangs far below the x-axis, zero-inflation has occurred.

To test whether a zero-inflated Poisson regression is better the zero-inflated function, encoded as *zeroinfl*, is used. The *summary* of this model displays the Pearson residuals, standard error, coefficients, and z test statistic. To compare the regular Poisson regression with the zero-inflated version a Vuong test can be used. This test determines if there is a statistically significant difference between the two models. The Vuong test allows for the comparison of non-nested models. Another possible item to look at is a plot of residuals, which display the difference between observed and predicted values.

For algorithms a Python environment is used, specifically on Jupyter notebook. There are many libraries that need to be imported including *numpy (*Harris and Millman 2020), *matplotlib* (Hunter 2007), *pandas* (McKinney 2010), *tensorflow* (Abadi et al. 2016), *seaborn* (Waskom et al. 2017), and *sklearn* (Pedregosa et al. 2011). These libraries contain necessary functions. Once those are loaded the dataset needs to be uploaded. Since this project involves datasets from R packages, one way to get the data on to Python is downloading the CSV (Comma Separated Values) version of the dataset and then using the read function coded as *read_csv*. The *describe* function is similar to the R *summary* function.

The data needs to go through a preprocessing phase. This includes insuring there are no missing values. Then, in order to separate the data into dependent and independent variables the columns must be reordered so the column for the dependent variable, or the variable being

predicted, is on one side or the other of the independent variables, which are the variables used to make the predictions. Once completed, using the *dataframe* function the variables can be divided into independent variables put into x and the dependent variable into y. Afterwards the categorical variables, such as gender, location, and vehicle body, are changed into numbers. For example, instead of using male or female the program will read a 0 or 1. For the ANN it is necessary to convert the count into a categorical response instead of simply integers. The number of categories or bins should be matched with the possible counts. Next the data then needs to be split into a training set and a testing set. These will be used to later train and test the algorithm. In this project twenty percent of the data is set aside for the testing set. The *transform* function can be used to standardize the scale of the dataset.

Once the data has been processed the algorithm itself can be created. For artificial neural networks (ANN) an additional library that needs to be imported is *keras* (Chollet 2015). This allows the ANN to be initialized using the *sequentia*l function. Then the input layer, hidden layer, and output layer are created. There can be as many hidden layers as desired. These use the dense function with a uniform initializer with a selected number of outputs for each layer. Activation functions are required for the various layers in order for the algorithm to determine what weights to use for each input. For the input layer and any hidden layer, a rectifier or *relu* activation function is used. The *relu* function has the benefit of being a simple computation that acts linear for positive values but produces a zero when values are negative ("Activation Functions in Neural Networks" n.d.). For the output layer a *softmax* activation function is used. The *softmax* function is a version of a *sigmoid* function which has an s shape and is used for multiclassification problems and produces probabilities ("Activation Functions in Neural Networks" n.d.). A summary of the layers can be useful to see what is happening with the

algorithm. The layers then need to be compiled with an *adam* optimizer, which "updates the weights during training and reduces loss" ("Implementation of Artificial Neural Network in Python" n.d.). A loss of categorical cross entropy which is used for categorical outputs as are necessary in this project. A performance metric based on accuracy is used.

Finally, the algorithm can be trained using a *fit* function. Using the training sets with a batch of 10, which is the number of sets that are used to train the algorithm at one time. The batch is runs through 15 times, as noted by epochs. The epochs are the steps or cycles that the batch goes through. Each of the fifteen cycles will fix weights assigned to the independent variable this will also display accuracy of each epoch. The *predict* function can then be utilized to create a prediction. Then the *evaluate* function can be used to find the model's total accuracy and well as loss. A confusion matrix as well as a classification report can be created for further assessment of the algorithm. In addition to accuracy, measures such as the kappa coefficient, sensitivity, specificity, precision, recall, and f1-scores can be computed. These extra measures are necessary since a high accuracy is not necessarily impressive in situations where most cases are in the same category. An example of this would be if the observed counts of insurance claims are mainly zeros with a minority of customers making one or more claims (Kuhn 2021).

A different algorithm that can be created after the data is processed is the k-nearest neighbor (KNN) algorithm. This algorithm is much simpler than the ANN. To create the KNN the *KNeighborsClassifier* function is used with the n neighbors set to five for the car insurance dataset since that is the normal k selection. The k is set to twenty-five for the biochemist dataset to maximize the accuracy of the KNN. Then to train the model the *fit* function using the training sets. Once completed the *score* function reports the accuracy of the model. For additional evaluation of the algorithm a confusion matrix in addition to a classification report can be used.

4. DATASETS

4.1 Car Insurance Dataset

The first dataset to be used involves the number of insurance claims made on car insurance policies. This dataset is contained within the package *insuranceData* and is called *dataCar*. This dataset contains 67856 observations (Wolny-Dominiak and Trzesiok 2014). These observations are the number of "one-year vehicle insurance policies taken out in 2004 or 2005" (Wolny-Dominiak and Trzesiok 2014). There are several variables recorded for each policy. These include:

- Veh_val - vehicle value which is in ten-thousands of dollars
- exposure which is between 0 and 1 and signifies the risk associated with the policy where one represents the highest risk
- clm- whether a claim occurred
- numclaims- number of claims
- claimcst0 - claim amounts in dollars
- veh_body - vehicle body type which covers thirteen different bodies such as Sedan or Coupe
- veh_age - vehicle age, which is put into one of four categories
- gender of the policy holder
- the area the policy is held which can be one of six separate areas
- agecat - age of policy holder is put into six categories (De Jong and Heller 2008).

The first six rows of data are displayed by the *head* function shown in Table 4.1.1.

Table 4.1.1

	veh_value	exposure	clm	numclaims	claimcst0	veh_body	veh_age	gender	area	agecat
1	1.06	0.3039	0	0	0	HBACK	3	F	C	2
2	1.03	0.6489	0	0	0	HBACK	2	F	A	4
3	3.26	0.5695	0	0	0	UTE	2	F	E	2
4	4.14	0.3176	0	0	0	STNWG	2	F	D	2
5	0.72	0.6489	0	0	0	HBACK	4	F	C	2
6	2.01	0.8542	0	0	0	HDTOP	3	M	C	4

The *summary* function in R and the *describe* function in Python provides more details on each variable and the totals. Both functions provide the minimums, maximums, and the quartiles of each variable. The *describe* function also details the mean and standard deviation of each variable which allows for a good sense of what is within the *dataCar* data set. The information from the *describe* function is displayed in Table 4.1.2.

Table 4.1.2

	VEH_VALUE	EXPOSURE	CLM	NUMCLAIMS	CLAIMCST0	VEH_AGE	AGECAT
COUNT	67856	67856	67856	67856	67856	67856	67856
MEAN	1.777	0.469	0.068	.073	137.270	2.674	3.485
STD	1.205	0.290	0.252	0.278	1056.298	1.068	1.425
MIN	0.000	0.003	0.000	0.000	0.000	1.000	1.000
25%	1.010	0.219	0.000	0.000	0.000	2.000	2.000
50%	1.500	0.446	0.000	0.000	0.000	3.000	3.000
75%	2.150	0.709	0.000	0.000	0.000	4.000	5.000
MAX	34.560	0.999	1.000	4.000	55922.130	4.000	6.000

4.2 Biochemist dataset

The second dataset to be used is about article production by graduate students from the field of biochemistry. The dataset is contained within the *pscl* package and is named *bioChemists*. This dataset is quite a bit smaller than the car insurance one with a sample of n=915 graduate students (Jackman 2020). The variables included in the data set are as follows:

- art - number of articles produced in the last three years of getting their Ph.Ds.
- fem - gender
- mar - marital status of the student
- kid5 - number of children under five the student has
- phd - prestige of the Ph. D department based on a scale from zero to five
- ment - number of articles produced by the student's mentor in the last three years.

The *head* function again shows how the variables are ordered as well as how they are coded. Table 4.2.1 has the first six students in the sampled, as produced by the *head* function.

Table 4.2.1

	art	fem	mar	kid5	phd	ment
1	0	Men	Married	0	2.52	7
2	0	Women	Single	0	2.05	6
3	0	Women	Single	0	3.75	6
4	0	Men	Married	1	1.18	3
5	0	Women	Single	0	3.75	26
6	0	Women	Married	2	3.59	2

To get more information about the entire set the R *summary* function or the Python *describe* function was used. Table 4.2.2 has the output from Python's *describe* function.

Table 4.2.2

	ART	**KID5**	**PHD**	**MENT**
COUNT	915	915	915	915
MEAN	1.693	0.495	3.103	8.767
STD	1.926	0.765	0.984	9.484
MIN	0.000	0.000	0.755	0.000
25%	0.000	0.000	2.260	3.000
50%	1.000	0.000	3.150	6.000
75%	2.000	1.000	3.920	12.000
MAX	19.000	3.000	4.620	77.000

5. RESULTS

5.1 Traditional Statistical Model

The Poisson regression produces a linear approximation of the data that can then be used to predict the count of the response variable. The *summary* function on the Poisson regression provides coefficients that are parameter estimates for each variable in addition to the standard error associated with each variable. Finally, the summary includes Wald z tests and p-values that are used to determine the statistical significance of each predictor variable. The car insurance dataset Poisson regression summary is contained in Table 5.1.1 and the biochemist dataset Poisson regression summary are in Table 5.1.2.

Table 5.1.1

Coefficients:

	ESTIMATE	STD. ERROR	Z VALUE	P-VALUE
(INTERCEPT)	-2.2754	0.3303	-6.888	< 0.0001
VEH_VALUE	0.0285	0.0169	1.683	0.0924
EXPOSURE	1.8024	0.0510	35.358	< 0.0001
VEH_BODYCONVT	-1.7535	0.6687	-2.622	0.0087
VEH_BODYCOUPE	-0.5594	0.3374	-1.658	0.0973
VEH_BODYHBACK	-0.9825	0.3186	-3.084	0.0020
VEH_BODYHDTOP	-0.8499	0.3278	-2.593	0.0095
VEH_BODYMCARA	-0.3988	0.4096	-0.974	0.3301
VEH_BODYMIBUS	-1.0283	0.3502	-2.936	0.0033
VEH_BODYPANVN	-0.8803	0.3392	-2.595	0.0095
VEH_BODYRDSTR	-0.6217	0.6601	-0.942	0.3463
VEH_BODYSEDAN	-0.9336	0.3180	-2.936	0.0033
VEH_BODYSTNWG	-0.9336	0.3181	-2.935	0.0033
VEH_BODYTRUCK	-0.9822	0.3284	-2.991	0.0028
VEH_BODYUTE	-1.1400	0.3221	-3.539	0.0004
VEH_AGE	-0.0437	0.0179	-2.439	0.0147
GENDERM	-0.0239	0.0301	-0.796	0.4262
AREAB	0.0506	0.0428	1.183	0.2369
AREAC	0.0045	0.0390	0.115	0.9086
AREAD	-0.1133	0.0530	-2.140	0.0324
AREAE	-0.0290	0.0579	-0.501	0.6167
AREAF	0.0660	0.0662	0.997	0.3187
AGECAT	-0.0880	0.0103	-8.578	< 0.0001

Table 5.1.2

Coefficients:

	ESTIMATE	STD. ERROR	Z VALUE	P-VALUE
(INTERCEPT)	0.3046	0.1030	2.958	0.0031
FEMWOMEN	-0.2246	0.0546	-4.112	< 0.0001
MARMARRIED	0.1552	0.0614	2.529	0.0114
KID5	-0.1849	0.0401	-4.607	< 0.0001
PHD	0.0128	0.0264	0.486	0.6271
MENT	0.0255	0.0020	12.733	< 0.0001

For the car insurance Poisson regression model, using Table 5.1.1 it can be determined that the vehicle value, gender, and areas other than area D seem to have minimal effect on the model because their coefficients have such small magnitudes. The exposure shows the greatest impact on the Poisson regression. Using a significance level of $\alpha=0.05$ the following predictor variables are statistically significant: exposure, convertible body, hatchback body, hardtop body, minibus body, panel van body, sedan body, station wagon body, truck body, utility body, vehicle age, area D, and the age category. All the other variables were not statistically significant.

The biochemist dataset shows that the prestige of the university has the smallest effect on the model with the smallest coefficient. The largest coefficient in magnitude was assigned to gender. Again, using a significance level of $\alpha=0.05$ the variables that were significantly significant were gender, marital status, number of children under five, and the number of papers written by the student's mentor in the last three years. The only variable that would not be considered statistically significant was the prestige of the university.

To evaluate the Poisson regression an analysis of deviance table can be used as well as a goodness of fit test. The analysis of deviance table that displays a likelihood ratio chi-square test statistic degrees of freedom as well as the p-value for the variables. The analysis of deviance table for car insurance data is in Table 5.1.3 and the analysis of deviance table for biochemist dataset is in Table 5.1.4.

Table 5.1.3

Analysis of Deviance Table (Type II tests)

Response: numclaims

	LR CHISQ	**DF**	**P-VALUE**
VEH_VALUE	2.72	1	0.0994
EXPOSURE	1314.38	1	< 0.0001
VEH_BODY	37.88	12	0.0002
VEH_AGE	6.00	1	0.0143
GENDER	0.63	1	0.4259
AREA	11.13	5	0.0488
AGECAT	73.82	1	< 0.0001

Table 5.1.4

Analysis of Deviance Table (Type II tests)

Response: art

	LR CHISQ	DF	PR(>CHISQ)
FEM (GENDER)	17.075	1	< 0.0001
MAR	6.431	1	0.0112
KID5	22.082	1	< 0.0001
PHD	0.236	1	0.6270
MENT	131.868	1	< 0.0001

From Table 5.1.3 with a significance level of $\alpha=0.05$ from the car insurance data the exposure, vehicle body, vehicle age, area, and age category are statistically significant with the number of claims made.

The biochemist data set has p-values less than the significance level of $\alpha=0.05$ indicating statistical significance for gender, marital status, children under five, and the number of articles produced by the student's mentor. This leaves only the prestige of the university not significantly associated with the number of articles produced.

The goodness of fit test has a likelihood ratio using the chi-squared test as well as a graph that shows the observed counts versus predicted counts. The car insurance data was found to have a chi-square test statistic of 104.5812 with three degrees of freedom and a p-value of less than 0.0001. The biochemist dataset resulted in a chi-square test statistic of 296.3715 with

thirteen degrees of freedom and a p-value of less than 0.0001. The goodness of fit graph for the car insurance data is Figure 5.1.1 and Figure 5.1.2 is the goodness of fit graph for the biochemist data. These graphs are known as hanging rootograms. The bar for the zero-count extending far below the horizontal axis indicates that there are many more zeros in the observed data than the Poisson regression model predicts. This is known as zero-inflation and can be addressed with a zero-inflated Poisson regression model.

Figure 5.1.1

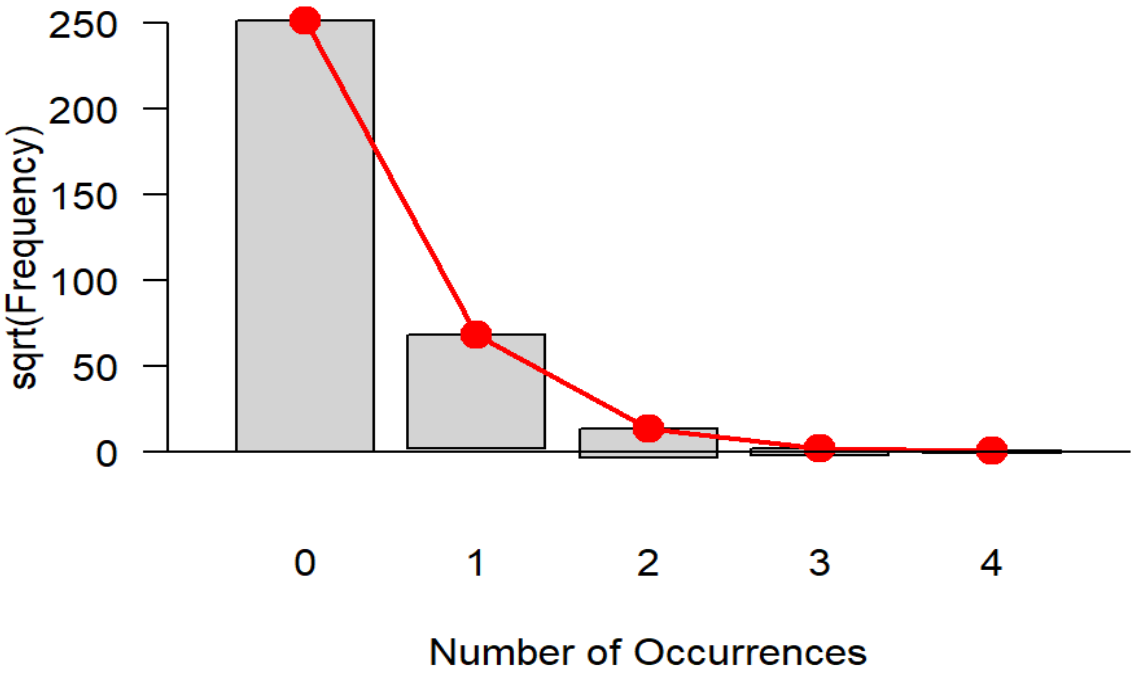
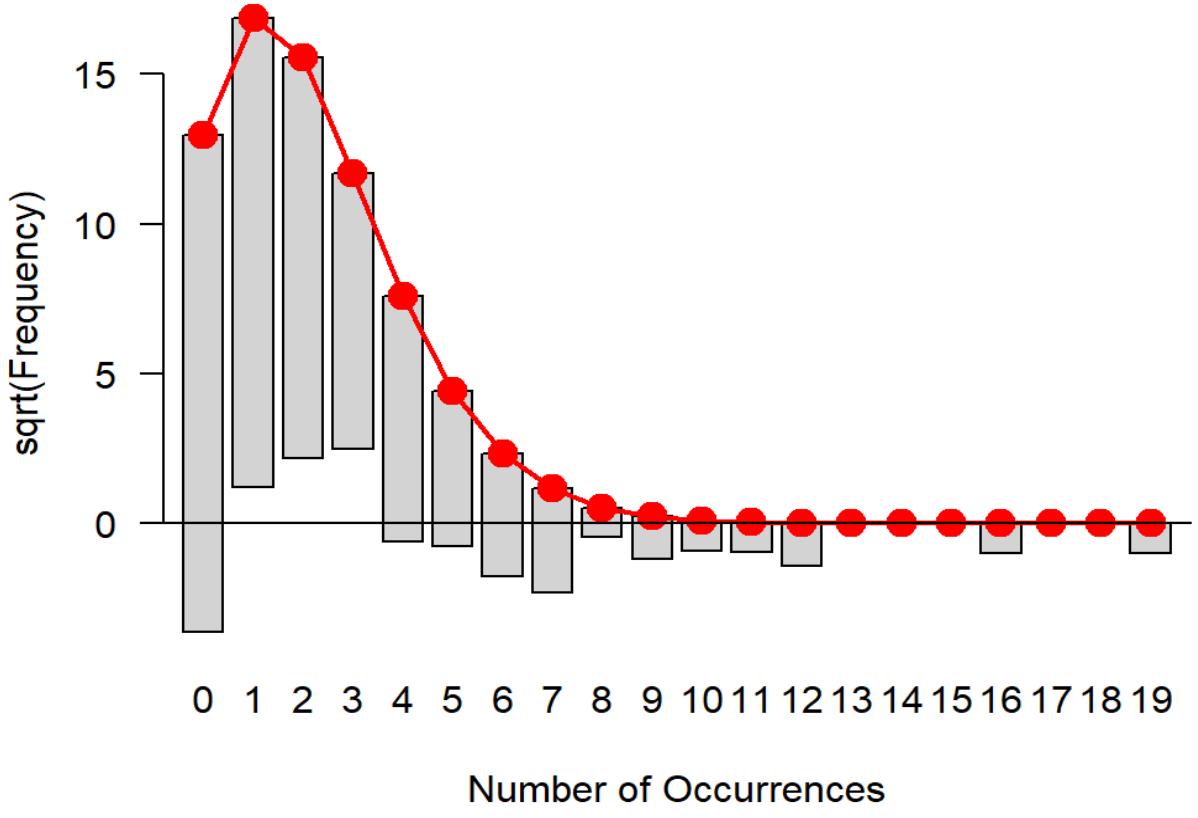


Figure 5.1.2



The goodness of fit test for both datasets return incredibly small p-values. This means even at a significance level of $\alpha=0.01$ the null hypothesis that the Poisson regression model fits the data is rejected. Thus, the data does not fit the model. Though Figure 5.1.1, which depicts a hanging rootogram for the car insurance data does not visually show a dramatic excess or lack of any number of occurrences that would be expected of a model that is a poor fit. The biochemist dataset's hanging rootogram in Figure 5.1.2 does show where the data does not match the data,

including an abundance of zeros the model leaves unaccounted for. This suggests that the zero-inflated Poisson regression may be a better model to use for the biochemist dataset.

The results of zero-inflated Poisson regression for car insurance data are in Table 5.1.5. For the biochemist dataset the zero-inflated Poisson regression is in Table 5.1.6.

Table 5.1.5

Count model coefficients (poisson with log link):

	ESTIMATE	**STD. ERROR**	**Z VALUE**	**P-VALUE**
(INTERCEPT)	-0.8545	0.5474	-1.561	0.1184
VEH_VALUE	0.0520	0.0317	1.639	0.1013
EXPOSURE	0.7417	0.1462	5.073	< 0.0001
VEH_BODYCONVT	-1.7893	1.0250	-1.746	0.0809
VEH_BODYCOUPE	-0.9627	0.5273	-1.826	0.0679
VEH_BODYHBACK	-1.4388	0.5037	-2.857	0.0043
VEH_BODYHDTOP	-1.2496	0.5147	-2.428	0.0152
VEH_BODYMCARA	-0.9268	0.6042	-1.534	0.1251
VEH_BODYMIBUS	-1.2420	0.5455	-2.277	0.0228
VEH_BODYPANVN	-1.1955	0.5240	-2.282	0.0225
VEH_BODYRDSTR	0.0927	1.1387	0.081	0.9351
VEH_BODYSEDAN	-1.3439	0.5027	-2.673	0.0075
VEH_BODYSTNWG	-1.3254	0.5054	-2.623	0.0087
VEH_BODYTRUCK	-1.2212	0.5139	-2.376	0.0175
VEH_BODYUTE	-1.4859	0.5078	-2.926	0.0034
VEH_AGE	-0.0895	0.0273	-3.279	0.0010
GENDERM	-0.0569	0.0408	-1.395	0.1630
AREAB	-0.0359	0.0603	-0.596	0.5512
AREAC	-0.0933	0.0563	-1.657	0.0974
AREAD	-0.2269	0.0744	-3.048	0.0023
AREAE	-0.1518	0.0809	-1.876	0.0607
AREAF	-0.0696	0.0912	-0.763	0.4454
AGECAT	-0.0711	0.0142	-5.011	< 0.0001

Table 5.1.6

Count model coefficients (poisson with log link):

	ESTIMATE	STD. ERROR	Z VALUE	P-VALUE
(INTERCEPT)	0.6408	0.1213	5.283	< 0.0001
FEMWOMEN	-0.2091	0.0634	-3.299	0.0010
MARMARRIED	0.1038	0.0711	1.459	0.1446
KID5	-0.1433	0.0474	-3.022	0.0025
PHD	-0.0062	0.0310	-0.199	0.8424
MENT	0.0181	0.0023	7.888	< 0.0001

For the car insurance data, the zero-inflated Poisson features the largest coefficient in magnitude as connected with a convertible body with a coefficient of -1.7893. Using the p-values and a significance level of $\alpha=0.05$ the variables that are statistically significant are exposure, hatch back body, hard top body, minibus body, panel van body, sedan body, station wagon body, truck body, utility body, vehicle age, area D, and age categories.

The biochemist dataset did not show a large shift in coefficients under the zero-inflated Poisson regression. The gender still has the largest coefficient in magnitude. The p-values for gender, children under five, and number of articles produced by the student's mentor imply that they are statistically significant when using a level of $\alpha=0.05$. Marital status as well as prestige of university are found to be non-significant with the number of articles produced. The marital status being no longer statistically significant is a change from the Poisson regression.

The analysis of deviance test can be performed on the zero-inflated Poisson regression as well. The analysis of deviance table for the car insurance dataset is located in Table 5.1.7. Table 5.1.8 contains the analysis of deviance table for the biochemist dataset.

Table 5.1.7

Analysis of Deviance Table (Type II tests)

Response: numclaims

	LR CHISQ	**DF**	**P-VALUE**
VEH_VALUE	2.6862	1	0.1012
EXPOSURE	25.7409	1	< 0.0001
VEH_BODY	26.7675	12	0.0083
VEH_AGE	10.7521	1	0.0010
GENDER	1.9479	1	0.1628
AREA	11.4473	5	0.0432
AGECAT	25.1170	1	< 0.0001

Table 5.1.8

Analysis of Deviance Table (Type II tests)

Response: art

	LR CHISQ	DF	PR(>CHISQ)
FEM (GENDER)	10.8805	1	0.0010
MAR	2.1287	1	0.1446
KID5	9.1310	1	0.0025
PHD	0.0395	1	0.8424
MENT	131.868	1	< 0.0001

Under a significance level of $\alpha=0.05$ the p-values for the car insurance dataset determine the same factors of exposure, vehicle body, vehicle age, area, and age category are statistically significant as in the Poisson regression. The zero-inflated Poisson regression for the biochemist data does feature a change. The Poisson regression of the biochemist data has marital status statistically significant to articles produced, but the zero-inflated Poisson regression does not. Otherwise, the p-values indicate the same variables are statistically significant including gender, children under five, and the number of articles the students mentor produced.

A statistical test to compare the Poisson regression with the zero-inflated Poisson regression is called the *Vuong* test. The *Vuong* test can be used to compare two non-nested statistical models. This test uses a null hypothesis that says two models are indistinguishable. The Vuong test results for the car insurance data is in Table 5.1.9. For the biochemist dataset the Vuong test is in Table 5.1.10.

Table 5.1.9

Vuong Non-Nested Hypothesis Test-Statistic:

(test-statistic is asymptotically distributed N(0,1) under the

null that the models are indistinguishible)

	VUONG Z-STATISTIC	**H_A**	**P-VALUE**
RAW	-6.9885	model2 > model1	< 0.0001

Table 5.1.10

Vuong Non-Nested Hypothesis Test-Statistic:

(test-statistic is asymptotically distributed N(0,1) under the

null that the models are indistinguishible)

	VUONG Z-STATISTIC	**H_A**	**P-VALUE**
RAW	-4.1805	model2 > model1	< 0.0001

The car insurance data from Table 5.1.9 has a p-value less than .0001 so the null hypothesis that the models are indistinguishable is rejected with a significance level of $\alpha=0.05$.

Table 5.1.10 also shows a p-value less than .0001 for the biochemist dataset. The null is rejected under a significance level of $\alpha=0.05$. The improvement caused by the zero-inflated Poisson regression is mostly likely more important for the biochemist dataset since as shown in Figure 5.1.2 the hanging rootogram of the Poisson regression visually shows an excess of zeros. That same property was not displayed in Figure 5.1.1 for the car insurance Poisson regression.

Thus, the biochemist dataset was a prime example of when to use the zero-inflated Poisson regression.

5.2 Machine Learning

Algorithmic modeling techniques were also used. These results appear to be simpler though it is not always as evident why these models make the predictions that they do. To get some idea to what the artificial neural networks is performing we start with a summary of the layers of the ANNs. The summary displays what type of model was used to build the layers as well as what type of layer each is. This also shows the number of outputs for each layer and number of parameters each uses. Table 5.2.1 is the summary of layers for the car insurance data. The biochemist data layer summary is in Table 5.2.2.

Table 5.2.1

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param
#
=================================================================
 dense (Dense)               (None, 4)                 32

 dense_1 (Dense)             (None, 4)                 20

 dense_2 (Dense)             (None, 5)                 25

=================================================================
Total params: 77
Trainable params: 77
Non-trainable params: 0
```

Table 5.2.2

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Par
am #
=================================================================
=======
 dense (Dense)               (None, 4)                 24

 dense_1 (Dense)             (None, 4)                 20

 dense_2 (Dense)             (None, 15)                75

=================================================================
=======
Total params: 119
Trainable params: 119
Non-trainable params: 0
_________________________________________________________________
_______
```

These summary tables make it clear how many layers were used and how many ‘neurons’ are to be used in each step. In Table 5.2.1 for the car insurance data from the seven independent variables the algorithm puts them into four nodes or neurons. In this process the algorithm created thirty-two parameters or weights to use for calculating the next layer. In a similar way layer two has four neurons with twenty more parameters. The final layer has the necessary number of outputs to match the categories being predicted. For the car insurance claims there are five categories: one for each number from zero to four which was the range of claims a policy holder made. This layer added another twenty-five parameters for a total of seventy-seven parameters used for the model.

Table 5.2.2 displays the same information for the biochemist dataset. There were more parameters needed to create the model with a total of 119. Both the first and second layers had

four nodes, meanwhile the output layer had fifteen. This was because that was the number of distinct amounts of articles produced ranged from zero to nineteen. The reason that there was fewer than twenty nodes in the output layer was because not every possible count was observed.

After the layers are made and compiled and the algorithm is being trained, the accuracy and time to complete each run through while fixing the weights assigned to inputs is recorded. The loss produced by the *categorical_crossentropy* function at each step or epoch are also recorded. For the last step of the car insurance ANN there was a reported accuracy of 0.9316 with a loss of 0.2563. The accuracy did not change with the steps, but the loss decreased slightly. As the later epochs took more time, the slight decrease in error may not be worth running all the epochs. Since the biochemist dataset is much smaller the ANN took considerably less time. Although the training was quicker, the accuracy was a low 0.3087 with a high loss of 1.8519 in the last epoch.

To get an overall accuracy of the algorithm the *evaluate* function is used. This function returns the loss and accuracy of the algorithm using the testing sets. For the car insurance data, the accuracy is 0.9329 and the loss is 0.2529. That means the ANN correctly classifies the number of claims 93% of the time. The biochemist data was on the other end of the spectrum with an accuracy of only 0.2678 with a loss of 1.9191. These accuracies and losses did not undergo much change between each step in the training and the overall accuracy and loss found with the testing sets.

The k-nearest neighbor (KNN) algorithm produces accuracies for the algorithm using the *score* function with the testing sets. The car insurance KNN ended up with an accuracy of 0.9304. Meanwhile the accuracy of the KNN for the biochemist dataset was only 0.3115. The accuracies produced by the KNNs were very similar to those produced by the ANNs. The KNN

had a slightly lower accuracy than the ANN for the car insurance dataset. For the biochemist data the KNN accuracy was marginally larger than the ANN accuracy.

Other metrics can be used to discern the effectiveness of the algorithms. One mechanism to be used is a confusion matrix which show how many elements are correctly classified and how many are classified incorrectly. For the ANN, the car insurance dataset the confusion matrix is in Table 5.2.3 and the biochemist dataset confusion matrix is in Table 5.2.4. The car insurance dataset KNN confusion matrix is found in Table 5.2.5. Finally, the KNN confusion matrix for the biochemist dataset is in Table 5.2.6.

Table 5.2.3

	0	1	2	3	4
0	[[12661	0	0	0	0]
1	[850	0	0	0	0]
2	[55	0	0	0	0]
3	[5	0	0	0	0]
4	[1	0	0	0	0]]

Table 5.2.4

	0	1	2	3	4	5	6	7	8	9	10	11
0	[[49	0	0	0	0	0	0	0	0	0	0	0]
1	[46	0	0	0	0	0	0	0	0	0	0	0]
2	[35	0	0	0	0	0	0	0	0	0	0	0]
3	[26	0	0	0	0	0	0	0	0	0	0	0]
4	[14	0	0	0	0	0	0	0	0	0	0	0]
5	[4	0	0	0	0	0	0	0	0	0	0	0]
6	[2	0	0	0	0	0	0	0	0	0	0	0]
7	[3	0	0	0	0	0	0	0	0	0	0	0]
8	[1	0	0	0	0	0	0	0	0	0	0	0]
9	[1	0	0	0	0	0	0	0	0	0	0	0]
10	[1	0	0	0	0	0	0	0	0	0	0	0]
11	[1	0	0	0	0	0	0	0	0	0	0	0]]

Table 5.2.5

	0	1	2	3	4
0	[[12623	38	0	0	0]
1	[846	4	0	0	0]
2	[54	1	0	0	0]
3	[5	0	0	0	0]
4	[1	0	0	0	0]]

Table 5.2.6

	0	1	2	3	4	5	6	7	8	9	10	11
0	[[38	11	0	0	0	0	0	0	0	0	0	0]
1	[29	17	0	0	0	0	0	0	0	0	0	0]
2	[22	13	0	0	0	0	0	0	0	0	0	0]
3	[10	16	0	0	0	0	0	0	0	0	0	0]
4	[9	5	0	0	0	0	0	0	0	0	0	0]
5	[0	4	0	0	0	0	0	0	0	0	0	0]
6	[0	2	0	0	0	0	0	0	0	0	0	0]
7	[2	1	0	0	0	0	0	0	0	0	0	0]
8	[1	0	0	0	0	0	0	0	0	0	0	0]
9	[0	1	0	0	0	0	0	0	0	0	0	0]
10	[1	0	0	0	0	0	0	0	0	0	0	0]
11	[1	0	0	0	0	0	0	0	0	0	0	0]]

The confusion matrices show that the algorithms either only estimated zeros or only estimated zeros or ones. Thus, the high accuracy for the car insurance dataset may be misleading, since that accuracy is due to the unbalanced set being primarily zeros which were predicted accurately.

Another function that could be looked at is the *classification_report*, which returns the precision, the recall, f1-score, and support for each count in addition to averages of those categories. The precision is the portion of correctly identified as a count over the total identified as that count. The recall notes the comparison between correctly identified and those that should have been identified as the same count. The closer to one the better the f1-score, which is the harmonic mean of the precision and the recall. The support identifies how many of each count was found within the dataset. The ANN car insurance dataset is in Table 5.2.7. The biochemist dataset ANN classification report is in Table 5.2.8. Table 5.2.9 contains the KNN car insurance

dataset classification report. Table 5.2.10 is comprised of the classification report for the biochemist KNN.

Table 5.2.7

	precision	recall	f1-score	support
0	0.93	1.00	0.97	12661
1	0.00	0.00	0.00	850
2	0.00	0.00	0.00	55
3	0.00	0.00	0.00	5
4	0.00	0.00	0.00	1

Table 5.2.8

	precision	recall	f1-score	support
0	0.00	0.00	0.00	49
1	0.00	0.00	0.00	46
2	0.00	0.00	0.00	35
3	0.00	0.00	0.00	26
4	0.00	0.00	0.00	14
5	0.00	0.00	0.00	4
6	0.00	0.00	0.00	2
7	0.00	0.00	0.00	3
8	0.00	0.00	0.00	1
9	0.00	0.00	0.00	0
10	0.00	0.00	0.00	1
11	0.00	0.00	0.00	1
12	0.00	0.00	0.00	1
13	0.00	0.00	0.00	0
14	0.00	0.00	0.00	0

Table 5.2.9

	precision	recall	f1-score	support
0	0.93	1.00	0.96	12661
1	0.09	0.00	0.01	850
2	0.00	0.00	0.00	55
3	0.00	0.00	0.00	5
4	0.00	0.00	0.00	1

Table 5.2.10

	precision	recall	f1-score	support
0	0.34	0.78	0.47	49
1	0.24	0.37	0.29	46
2	0.00	0.00	0.00	35
3	0.00	0.00	0.00	26
4	0.00	0.00	0.00	14
5	0.00	0.00	0.00	4
6	0.00	0.00	0.00	2
7	0.00	0.00	0.00	3
8	0.00	0.00	0.00	1
10	0.00	0.00	0.00	1
11	0.00	0.00	0.00	1
12	0.00	0.00	0.00	1

6. CONCLUSION

There is no model that is the best method to use in all situations. Looking at the Poisson regression compared with the zero-inflated Poisson regression with the two datasets using the Vuong test shows that the two models are distinct. For the car insurance data in Table 5.1.9 the raw results point to the conclusion that the zero-inflated Poisson regression is better. We noticed from the hanging rootogram for the Poisson regression in Figure 5.1.1 made with the goodness of fit test that the number of claims closely matches the number that would be expected using the Poisson regression. Thus, despite the fact that the data set had a large number of zeros, there was not an overabundance of unaccounted for zeros.

The biochemist Vuong test found in Table 5.1.10 shows that the raw data resulted in a significant difference between the two models even using a significance level set at $\alpha=0.05$. These differences resulted in the zero-inflated model having a better fit for the data. This result is visually depicted using the biochemist Poisson regression rootogram in Figure 5.1.2. The graph shows a distinct overabundance of zeros implying a zero-inflated Poisson regression would do better.

The two machine learning algorithms had similar accuracies. This points to the fact that one algorithm may not be any better than the other, although the artificial neural network builds a more complex model than the k-nearest neighbor algorithm. This could impact the decision of which algorithm to use. Another factor against the ANN is that though it can be used to predict counts by separating them into categories this can be a complicated process and cause issues especially with a large variety of counts possible such as in the biochemist dataset. An ANN may be more suited for a binomial response variable such as will an insurance claim be made opposed to how many claims will be made.

The traditional statistical models and algorithmic models are difficult to compare with each other due to both having distinct methods of evaluation. Also, the different datasets produced wildly different results. For the car insurance dataset, the algorithms produced roughly 93% accuracy while the Poisson regression was determined not to fit the data with the goodness of fit test. Though the high accuracy may make the model seem better than it is, this is due to the ease of predicting zeros which are the majority of cases. Thus, the real cost of the mistakes could be high. On the other hand, the highest accuracy for the biochemist dataset came from the KNN at 31%. This poor accuracy suggests that the traditional statistical models might work better.

A reason for the division in which method works is that the car insurance dataset had over 67000 cases while the biochemist dataset only had slightly over 900 cases. For large datasets, traditional statistical tests will have high power so even quite small deviations from the null hypothesis will be deemed statistically significant. Thus, the machine learning algorithms might be preferable for large datasets. Meanwhile the biochemist dataset might not have a sufficient sample to be adequate for algorithmic modeling. This would be an issue for many biology studies because most biology datasets are not going to have thousands of subjects.

Another component is the range of counts possible the car insurance data had a maximum of four claims that could be made. The biochemist dataset ran from zero to nineteen articles published. This increased range might have caused to many categories to be needed for the algorithm. It is possible that if the counts were put into groups such as four or less in one, five to ten in another and so on to create less categories the algorithm may do better, but then the prediction would be less specific.

For future research a negative binomial regression could be used. Due to larger possible variance, it may simulate real world circumstances better than the Poisson regression. To

improve the algorithms the Synthetic Minority Technique (SMOTE) filter, which lowers bias by correcting imbalanced data, could be applied to the datasets (Skorikov et al. 2021). Other possibilities would be to try other types of machine learning algorithms. One that might be beneficial to compare is a decision tree network sometimes referred to as random forests. This network can be used for classification and regression which can be useful for predicting count data. Employing more datasets may also be illuminating. Datasets that contain different elements, such as a dataset that involves repeated measurements over a period of time or space. This type of data would require a generalized linear mixed model with random effects.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Viégas, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2016). "TensorFlow: Large-scale machine learning on heterogeneous systems," Software available from tensorflow.org.

Breiman, L. (2001). "Statistical Modeling: Two Cultures," *Statistical Science*, 16, 199-231.

Bolker, B. M. (2008). *Ecological Models and Data in R*, Princeton, NJ: Princeton University Press.

Chollet, F. (2015). "Keras," Available at: https://github.com/fchollet/keras.

Fox, J., and Weisberg, S. (2019). *An R Companion to Applied Regression* (3rd ed.), Thousand Oaks CA: Sage.

Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., Fernández del Río, J., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). "Array programming with NumPy," *Nature*, 585, 357–362.

Hogg, R. V., Tanis, E. A., and Zimmerman, D. L. (2015). *Probability and Statistical Inference* (9th ed.), Pearson Education, Inc., Upper Saddle River, NJ.

Hunter, J. D. (2007). "Matplotlib: A 2D Graphics Environment," *Computing in Science & Engineering*, 9, 90-95.

"Implementation of Artificial Neural Network in Python," (n.d.), *ML Tut*. Available at https://www.mltut.com/implementation-of-artificial-neural-network-in-python/

Jackman, S. (2020). "pscl: Classes and Methods for R Developed in the Political Science Computational Laboratory," *United States Studies Centre, University of Sydney*. Sydney, New South Wales, Australia. R package version 1.5.5. Available at https://github.com/atahk/pscl/

James G., Witten D., Hastie T., and Tibshirani R. (2021). *Linear Regression. In: An Introduction to Statistical Learning*, Springer Texts in Statistics. Springer, New York, NY.

Kleiber, C., and Zeileis, A. (2016). "Visualizing Count Data Regressions Using Rootograms," *The American Statistician*, 70, 296-303.

Kuhn, M. (2021). "caret: Classification and Regression Training," R package version 60-90.

Loeys, T., Moerkerke, B., De Smet, O. and Buysse, A. (2012). “The Analysis of Zero-Inflated Count Data: Beyond Zero-Inflated Poisson Regression,*” British Journal of Mathematical and Statistical Psychology*, 65, 163-180.

McKinney, W. (2010). “Data Structures for Statistical Computing in Python,” In *Proceedings of the 9th Python in Science Conference*, 445, 51–56.

Meyer, D., Zeileis, A., and Hornik, K. (2021). “vcd: Visualizing Categorical Data,” R package version 1.4-9.

Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., and Dubourg, V. (2011). “Scikit-learn: Machine learning in Python.” *Journal of Machine Learning Research*, 12, 2825–2830.

Pruim, R., Kaplan, D. T., and Horton, N. J. (2017) “The mosaic Package: Helping Students to 'Think with Data' Using R,” *The R Journal*, 9, 77-102.

Simsek, S., Genc, O., Albizri, A., Dinc, S., and Gonen, B. (2020) “Artificial Neural Network Incorporated Decision Support Tool for Point Velocity Prediction,” *Journal of Business Analytics*, 3, 67-78.

Skorikov, M., Hussain, M. A., Khan, M. R., Akbar, M. K., Momen, S., Mohammed, N., and Nashin, T. (2020). "Prediction of Absenteeism at Work using Data Mining Techniques," *2020 5th International Conference on Information Technology Research (ICITR)*, pp. 1-6.

Tiwari, S. (2020), "Activation Functions in Neural Networks," *Geeks for Geeks*. Available at https://www.geeksforgeeks.org/activation-functions-neural-networks/eksforGeeks

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S*. (4th ed.), Springer, New York.

Waskom, M., Botvinnik, O., O'Kane, D., Hobson, P., Lukauskas, S., Gemperline, D. C., Augspurger, T., Halchenko, Y., Cole, J. B., Warmenhoven, J., de Ruiter, J., Pye, C., Hoyer, S., Vanderplas, J., Villalba, S., Kunter, G., Quintero, E., Bachant, P., Martin, M., Meyer, K., Miles, A., Ram, Y., Yarkoni, T., Williams, M. L., Evans, C., Fitzgerald, C., Fonnesbeck, C., Lee, A., and Qalieh, A. (2017). *mwaskom/seaborn: v0.8.1 (September 2017)*, Zenodo. Available at: https://doi.org/10.5281/zenodo.883859.

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., Takahashi, K., Vaughan, D., Wilke, C., Woo, K., and Yutani, H. (2019). "Welcome to the tidyverse." *Journal of Open Source Software*, 4, 1686.

Wolny-Dominiak, A., and Trzesiok, M. (2014). “insuranceData: A Collection of Insurance Datasets Useful in Risk Classification in Non-life Insurance,” R package version 1.0. Available at https://CRAN.R-project.org/package=insuranceData

Zeileis, A., and Kleiber, C. (2020). “countreg: Count Data Regression,” R package version 0.2-1. Available at http://R-Forge.R-project.org/projects/countreg/

Zeileis, A., Kleiber, C., and Jackman, S. (2008). “Regression Models for Count Data in R,” *Journal of Statistical Software*, 27, 1-25.

“Zero-Inflated Poisson Regression,” (n.d.). *UCLA: Statistical Consulting Group*, Available at https://stats.oarc.ucla.edu/r/dae/zip/

APPENDIX

A. R Code for Car Insurance Dataset

```
# Models Predictions on Car Insurance Data

# Load necessary packages and data set

require(tidyverse)

require(mosaic)

require(MASS)

install.packages("countreg",repos="http://R-Forge.R-project.org")

require(countreg)

require(vcd)

require(pscl)

require(car)

require(insuranceData)



#Looking at a summary of the data set

data("dataCar")

summary(dataCar)

head(dataCar)
```

```
# Poisson Regression

mod_Poisson <-
glm(numclaims~veh_value+exposure+veh_body+veh_age+gender+area+agecat,
family=poisson, data=dataCar)

summary(mod_Poisson)


#ANOVA analysis of Poisson regression

Anova(mod_Poisson)


# Zero-Inflated Poisson Regression

mod_zip <-
zeroinfl(numclaims~veh_value+exposure+veh_body+veh_age+gender+area+agecat|veh_value+
exposure+veh_body+veh_age+gender+area+agecat, dist="poisson",data=dataCar)

summary(mod_zip)


# Compare Poisson with Zero-inflated Poisson

vuong(mod_Poisson,mod_zip)
```

```
Anova(mod_zip)



#Goodness of fit Test

gf.poisson <- goodfit(x=dataCar$numclaims, type="poisson")

summary(gf.poisson)

plot(gf.poisson)



#Residuals Plots

plot(residuals(mod_Poisson)~fitted(mod_Poisson))

plot(residuals(mod_zip)~ fitted(mod_zip))
```

B. R Code for Biochemist Dataset

```
# Model Predictions on articles produced by biochemist students

require(tidyverse)

require(mosaic)

require(MASS)

install.packages("countreg",repos="http://R-Forge.R-project.org")

require(countreg)

require(vcd)

require(pscl)

require(car)



#Looking at summary of data

data("bioChemists")

summary(bioChemists)

head(bioChemists)



# Poisson Regression

mod_Poissonbio <- glm(art~fem+mar+kid5+phd+ment, family=poisson, data=bioChemists)

summary(mod_Poissonbio)
```

```
#ANOVA analysis of poisson

Anova(mod_Poissonbio)


# Zero-Inflated Poisson Regression

mod_zipbio <- zeroinfl(art~fem+mar+kid5+phd+ment | fem+mar+kid5+phd+ment,

dist="poisson", data=bioChemists)

summary(mod_zipbio)


# Comparing the Poisson regression with Zero-inflated Poisson regression

vuong(mod_Poissonbio,mod_zipbio)

Anova(mod_zipbio)


#Goodness of fit Test

gf.poissonbio <- goodfit(x=bioChemists$art, type="poisson")

summary(gf.poissonbio)

plot(gf.poissonbio)


#Residuals Plots
```

```
plot(residuals(mod_Poissonbio)~fitted(mod_Poissonbio))

plot(residuals(mod_zipbio)~ fitted(mod_zipbio))
```

C. Python Code for Car Insurance Dataset ANN

```
# Useful Libraries to load

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import tensorflow as tf

import seaborn as sns

import keras

from keras.utils import np_utils



#Read the data

dataset = pd.read_csv(r'C:\Users\nurse\Downloads\car.csv')

dataset.head()

dataset.describe()



# Checking to see if there are null vales

dataset.isnull().sum().max()



#Reordering the columns
```

```
column_names=["clm", "claimcst0", "numclaims", "veh_value", "exposure", "veh_body",
"veh_age","gender", "area", "agecat", "_OBSTAT_"]

dataset=dataset.reindex(columns=column_names)

dataset.head()


# Separating the Data into X and Y

x = pd.DataFrame(dataset.iloc[:, 3:10].values)

y = pd.DataFrame(dataset.iloc[:, 2].values)

print (x)

print(y)


# Encoding categorical data - Gender

from sklearn.preprocessing import LabelEncoder, OneHotEncoder

labelencoder_x_2 = LabelEncoder()

x.loc[:, 4] = labelencoder_x_2.fit_transform(x.iloc[:, 4])


# Encoding categorical data - Car body

labelencoder_x_13 = LabelEncoder()

x.loc[:, 2] = labelencoder_x_13.fit_transform(x.iloc[:, 2])
```

```
# Encoding categorical data - Area

labelencoder_x_6 = LabelEncoder()

x.loc[:, 5] = labelencoder_x_6.fit_transform(x.iloc[:, 5])

print (x)



# Breaking y into categories

y=pd.cut(np.ravel(y),bins=5,labels=[0,1,2,3,4])

encoder=LabelEncoder()

encoder.fit(y)

y=encoder.transform(y)

y=np_utils.to_categorical(y)

print(y)



# Separate into training and test sets

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 0)
```

```
# Scaling the data

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

x_train = sc.fit_transform(x_train)

x_test = sc.transform(x_test)



#Other needed libraries

import keras

from keras.models import Sequential

from keras.layers import Dense



# Initialize the Artificial Neural Network

classifier = Sequential()



#Create input, hidden, and output layers

classifier.add(Dense(4, kernel_initializer='uniform', activation = 'relu', input_dim = 7))

classifier.add(Dense(4, kernel_initializer='uniform', activation = 'relu'))

classifier.add(Dense(5,kernel_initializer='uniform', activation = 'softmax'))
```

```
classifier.summary()


# Compiling the model

classifier.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])


#Training the model

classifier.fit(x_train, y_train, batch_size = 10, epochs = 15)


# Predictions to test accuracy

y_pred = classifier.predict(x_test)


y_pred=(y_pred>0.5)


print(y_pred)


# Evaluate the model

results=classifier.evaluate(x_test, y_test,batch_size=10)
```

```
print('results')


# Create a confusion matrix

from sklearn.metrics import confusion_matrix, accuracy_score

cm = confusion_matrix(y_test.argmax(axis=1), y_pred.argmax(axis=1))

print(cm)

accuracy_score(y_test,y_pred)


# Create Classification Report

from sklearn.metrics import classification_report


print(classification_report(y_test,y_pred))
```

D. Code for Car Insurance Dataset KNN

```
# Import necessary modules

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model_selection import train_test_split

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import tensorflow as tf

import seaborn as sns



#Read the data

dataset = pd.read_csv(r'C:\Users\nurse\Downloads\car.csv')

dataset.head()

dataset.describe()



# Checking to see if there are null vales

dataset.isnull().sum().max()



#Reordering the columns
```

```
column_names=["clm", "claimcst0", "numclaims", "veh_value", "exposure", "veh_body",
"veh_age","gender", "area", "agecat", "_OBSTAT_"]
dataset=dataset.reindex(columns=column_names)
dataset.head()


# Separating the Data into X and Y
x = pd.DataFrame(dataset.iloc[:, 3:10].values)
y = pd.DataFrame(dataset.iloc[:, 2].values)
print (x)
print(y)


# Encoding categorical data - Gender
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_x_2 = LabelEncoder()
x.loc[:, 4] = labelencoder_x_2.fit_transform(x.iloc[:, 4])


# Encoding categorical data - Car body
labelencoder_x_13 = LabelEncoder()
x.loc[:, 2] = labelencoder_x_13.fit_transform(x.iloc[:, 2])
```

```
# Encoding categorical data - Area

labelencoder_x_6 = LabelEncoder()

x.loc[:, 5] = labelencoder_x_6.fit_transform(x.iloc[:, 5])

print (x)



# Separate into training and test sets

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 0)



# Scaling the data

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

x_train = sc.fit_transform(x_train)

x_test = sc.transform(x_test)



# Create KNN

classifier = KNeighborsClassifier(n_neighbors=5)
```

```
# Train the KNN

classifier.fit(x_train, np.ravel(y_train) )



# Get the accuracy of the KNN

print (classifier.score(x_test, y_test))



# Create a confusion matrix

from sklearn.metrics import confusion_matrix, accuracy_score

cm = confusion_matrix(y_test, y_pred)

print(cm)

accuracy_score(y_test,y_pred)



# Create Classification Report

from sklearn.metrics import classification_report



print(classification_report(y_test,y_pred))
```

E. Code for Biochemist Dataset ANN

```
#Loading Important libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import tensorflow as tf

import seaborn as sns

import keras

from keras.utils import np_utils



#Read the data

dataset = pd.read_csv(r'C:\Users\nurse\Downloads\dataset-24904.csv')

dataset.head()

dataset.describe()



# Checking to see if there are null vales

dataset.isnull().sum().max()



# Separating the Data into X and Y
```

```
x = pd.DataFrame(dataset.iloc[:, 1:6].values)

y = pd.DataFrame(dataset.iloc[:, 0].values)

print (x)

print(y)



# Encoding categorical data - Marital Status

from sklearn.preprocessing import LabelEncoder, OneHotEncoder

labelencoder_x_2 = LabelEncoder()

x.loc[:, 1] = labelencoder_x_2.fit_transform(x.iloc[:, 1])



# Encoding categorical data - Gender

from sklearn.preprocessing import LabelEncoder, OneHotEncoder

labelencoder_x_2 = LabelEncoder()

x.loc[:, 0] = labelencoder_x_2.fit_transform(x.iloc[:, 0])



print(x)



# Breaking y into categories
```

```
y=pd.cut(np.ravel(y),bins=20)
encoder=LabelEncoder()
encoder.fit(y)
y=encoder.transform(y)
y=np_utils.to_categorical(y)
print(y)

# Separate into training and test sets
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.20, random_state = 0)

# Scaling the data
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

#Other needed libraries
import keras
from keras.models import Sequential
from keras.layers import Dense

# Initialize the Artificial Neural Network
classifier = Sequential()

#Create input, hidden, and output layers
classifier.add(Dense(4, kernel_initializer='uniform', activation = 'relu', input_dim = 5))
classifier.add(Dense(4, kernel_initializer='uniform', activation = 'relu'))
```

```
classifier.add(Dense(15,kernel_initializer='uniform', activation='softmax'))

classifier.summary()

# Compile the ANN
classifier.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the ANN
classifier.fit(x_train, y_train, batch_size = 10, epochs = 10)

# Predictions to test accuracy
y_pred = classifier.predict(x_test)
y_pred = (y_pred > 0.5)
print(y_pred)

# Get the accuracy of the ANN
results=classifier.evaluate(x_test, y_test,batch_size=10)

print('results')

# Create a confusion matrix
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test.argmax(axis=1), y_pred.argmax(axis=1))
print(cm)
accuracy_score(y_test,y_pred)

# Create Classification Report
from sklearn.metrics import classification_report
```

```
print(classification_report(y_test,y_pred))
```

F. Code for Biochemist Dataset KNN

```
# Import necessary modules

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model_selection import train_test_split

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import tensorflow as tf

import seaborn as sns



#Read the data

dataset = pd.read_csv(r'C:\Users\nurse\Downloads\dataset-24904.csv')

dataset.head()

dataset.describe()



# Separating the Data into X and Y

x = pd.DataFrame(dataset.iloc[:, 1:6].values)

y = pd.DataFrame(dataset.iloc[:, 0].values)

print (x)
```

```
print(y)


# Encoding categorical data - Marital Status
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_x_2 = LabelEncoder()
x.loc[:, 1] = labelencoder_x_2.fit_transform(x.iloc[:, 1])


# Encoding categorical data - Gender
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_x_2 = LabelEncoder()
x.loc[:, 0] = labelencoder_x_2.fit_transform(x.iloc[:, 0])


print(x)


# Separate into training and test sets
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.20, random_state = 0)
```

```
# Scaling the data

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

x_train = sc.fit_transform(x_train)

x_test = sc.transform(x_test)



# Create KNN

classifier = KNeighborsClassifier(n_neighbors=25)



# Train the KNN

classifier.fit(x_train, np.ravel(y_train) )



# Get the accuracy of the KNN

print (classifier.score(x_test, y_test))



# Create a confusion matrix

from sklearn.metrics import confusion_matrix, accuracy_score

cm = confusion_matrix(y_test, y_pred)
```

```
print(cm)

accuracy_score(y_test,y_pred)



# Create Classification Report

from sklearn.metrics import classification_report



print(classification_report(y_test,y_pred))
```