Machine learning for advanced analytics

Tyler Shanu-Wilson

17003523

25/04/2020

A statement of originality and authenticity

I declare that the following research is my own work. All secondary sources are fully acknowledged and referenced. All the research detailed below is fully in compliance with the Liverpool Hope University research ethics policy and involves no human participants and no risk to the researcher.

Signed

……………………………………………………….

Date

25/04/2020

Acknowledgements

I would like to thank my research project tutor, Neil Buckley, for the all the support and advice throughout this year. I would also like to thank the rest of the department of mathematics and engineering at Liverpool Hope University for teaching and provision of the facilities provided in order for me to achieve my best work. Lastly, I would like to thank my family and friends for the constant support and motivation.

Abstract

Linear regression is a statistical method of modelling how one or more variables (dependant and independent) relate to each other. It can also be used as a machine learning algorithm. In simple linear regression, two variables are used to derive a formula to model the relationship between them, known as the regressor or linear function. The function can be represented on a graph of the two variables as the equation of a straight line that predicts the dependent variable in the most accurate way possible. Given a value of x you can use the function to predict y.

Contents

1.0 Introduction

1.1 background

The following research is based on machine learning and how it can be used to predict analytics. Machine learning is a subset of artificial intelligence. The definition of machine learning is the idea that algorithms can learn or adapt from sets of data and identify patterns. A.I (artificial intelligence) refers to a machines ability to perform a task in an intelligent way, such as a human would.

AI and machine learning can be used to perform tasks more efficiently and better than a human could. For example, in 1959 Arthur Samuels solidified his research in machine learning and was able to create an algorithm that can beat humans at the game of checkers. Machine learning has many other uses such as weather forecasting and stock market analysis.

Machine learning can be used to make things safer. An example of this is self-driving cars. Assuming the technology is perfected, it will use data such as the distance from an object to predict and prevent collisions by carrying out evasive actions that would prevent or minimise injury to passengers and/or pedestrians. The advantage to this over a human driving the car is the data can be processed a lot quicker than a human brain.

Another use that is important to people's lives is to personalise user experience on apps or websites by showing suggestions based on what they have been browsing or viewing previously. This is used on apps such as Netflix where suggested films are shown based on what other films you have watched and also on competing social media sites to capture the majority share of the users screen time.

In general, machine learning is important because a machine can learn from sets of data and use it to predict other data in a way that would be too complicated or less efficient for a human to do themselves. This has endless uses, but this research will focus mainly on predicting analytics, specifically for sports.

1.2 aims and objectives

The focus of this research is on the use of machine learning and how it can be used for advanced analytics. The main aim was to test how accurate the single variable linear regression algorithm is at predicting analytics. To find out how this works a machine learning algorithm was created for use with data from sports analytics to conclude how accurate and efficient it is at analysing data patterns and predicting further data. Listed below are the objectives of my research:

- **1.3 methodological description**- a brief description of the methodology used to investigate machine learning for advanced analytics
- **2 literature review**- an analysis of other research material on the same subject
- **3 Methodology and Research Design**- a detailed description of how the research has been carried out
- **4 results**- a review and discussion of the findings from the research carried out
- **5 conclusion**- a summary and analysis of the research as a whole and details of further research to be carried out in the future

1.3 Methodological description

To programme a linear regression algorithm in Jupyter notebook to train a dataset of sports analytics in order to derive a function. Secondly the algorithm should then be able to predict the values of a variable based on the function and the values of another variable. Finally, the accuracy of the algorithm should be tested in various ways in order to compare with the work of others. The methods of measuring accuracy to be used are:

- Root means squared error
- Root squared (r squared)
- Absolute error
- Absolute percentage error

2.0 Literature Review

2.1 machine learning for stock market analysis

People use the stock market to invest and trade in stocks in order to turn a profit. If the value of stock increases, it can be sold for a profit. The risk associated with this is the stock market is extremely unpredictable and values of stocks can decrease unexpectedly causing investors to lose money. The key to successful trading on the stock market is being able to identify patterns in the value of stocks over time so traders can invest when they are cheap and sell for more money to maximise the profit.

A solution to the volatile nature of the stock market is to use machine learning to identify patterns more accurately than a human could. Assuming a machine learning algorithm has been given enough data, it will be able to make predictions of the value of a stock at any point in the future. With this information, investors will have a better idea of when to buy and sell stocks for maximum profit.

If a graph of stock prices over time was to be plotted, the dependent and independent variables can be identified in order to identify relationships between them (referred to as linear regression in statistics). A classifier in statistics and machine learning, which is constructed from the variables, is the algorithm or mathematical function used to classify input data to a category. The classifier must be trained with data. The variables used for the classifier are essential to its accuracy and the more training data used, the more accurate the classifier. In this case the longer the period of time that stock prices have been recorded and the more information there is about the stock, the more accurate the classifier therefore the prediction of the future stock prices will be more accurate. Using supervised machine learning, when training the algorithm, the values that the classifier is predicting can be corrected to improve the accuracy.

The attributes most commonly used for graphs of stocks are: Open (Opening price of Stock), High (Highest price possible at an instance of time), Low (Lowest price possible at an instance of time), Close (Closing price of stock), Volume (Total times traded during a day) *(Pahwa and Agarwal, 2019)*. The feature "close" can be predicted using the other attributes and plotting a graph called an OHLCV (Open, High, Low, Close, Volume) graph in stock analysis. The percentage change of stock values can be calculated using the formula $HL_{PCT} = \frac{high - low}{close} \times 100$ which is used to obtain the relevant information from high, low and closed but reduced to one feature and can be used to determine the shape of the graph. We can also derive another feature PCT_{change} calculated using $PCT_{change} = \frac{close - open}{open} \times 100$. Another feature used to form the classifier is volume. It is important that the correct features are used to form the classifier and that they are calculated correctly to ensure the classifier is efficient.

The data extracted from the attributes is then pre-processed and split into test and train data. This data is then input into the classifier. For simple problems such as this, the linear regression classifier is sufficient as it works by analysing the key features to predict relations between variables. This is a form of supervised machine learning. The way this works is, the classifier is trained until it learns the patterns of features and how they relate to each label. It can then follow these same patterns and predict future labels. To test this, certain features that you already know the label for, (training data) are input into the classifier and see if the results match up. The accuracy of this is very important and an accuracy of at least 95% is expected for reliable results.

In one study, the researchers were able to follow these principals to predict future stock values. They were then able to plot these results on a graph *(fig.2.1).* In conclusion, this is a good method for this application as they were able to achieve an accuracy of 97.7% which is very accurate for a simple algorithm as it is higher than 95%. Similar results have also been achieved in other studies proving the method is reliable.

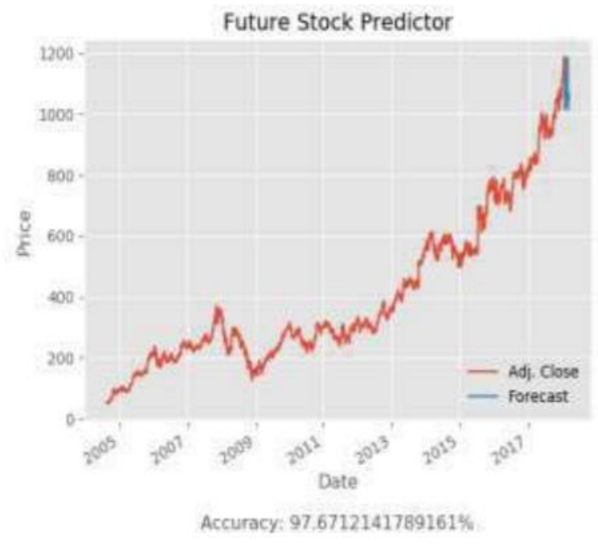


Fig.2.1 (Pahwa and Agarwal, 2019)

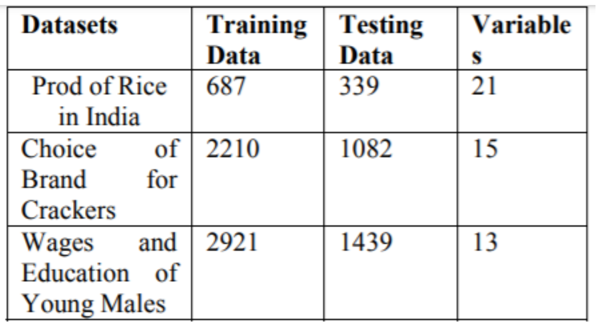
2.2 most accurate machine learning algorithms

There are three types of machine learning algorithm: supervised learning, unsupervised learning and reinforcement learning. The main principle of unsupervised learning is to cluster labels into groups based on their variable values. This can be used to split data into groups that would otherwise not be considered as people don't usually notice the more complex patterns between data. A good use for this is to split customers into specific groups in order to target advertising to each group. One of the most commonly used unsupervised machine learning algorithms is K-means

In supervised learning, there is a dependent variable which is to be predicted from a set of independent variables using a function or classifier. This classifier can change as it is trained with data until an acceptable level of accuracy is achieved. Once this level of accuracy is achieved, the test data can be used with the algorithm for an accurate prediction of the dependent variable. The main difference between supervised and unsupervised is in supervised the data is split between training data and test data. This algorithm is most commonly used for analytics as you can train the algorithm until it is accurate enough for your test data. Examples of supervised learning algorithms are regression, decision tree and KNN (K nearest neighbours).

Reinforcement learning is where the algorithm uses trial and error in order to train itself to make specific decisions based on certain input data. One complex problem is split into smaller sub problems. The idea of cumulative reward is then used, which refers to the idea that the algorithm gets a certain reward for each decision and will try to earn the maximum total reward by making the correct decision under each circumstance. This means the algorithm will make decisions to earn a greater reward for the next decision even if it means not taking the maximum reward for the current decision. This is used for very complex problems such as certain business decisions. The most commonly used reinforcement learning algorithm is the Markov decision process as it utilizes dynamic programming. Artificial neural networks are and example of reinforcement learning.

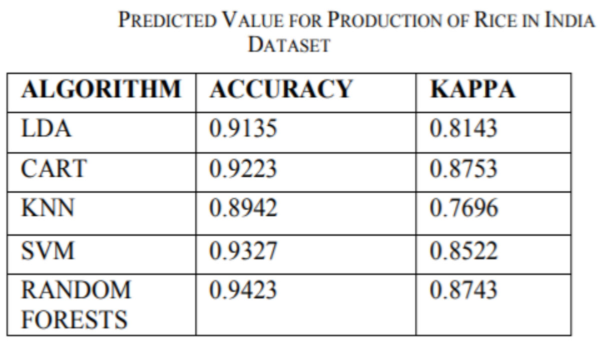
In one study *(Sasikala, Biju and Prashanth, 2017)*, the following data sets were used to find the accuracy of different machine learning algorithms: Production of Rice in India, Choice of Brand for Crackers, Wages and Education of Young Males. These datasets were used to calculate the accuracy of the LDA (Latent Dirichlet allocation), CART (Classification And Regression Trees), KNN (K Nearest Neighbours), SVM (Support Vector Machines) and random forests machine learning algorithms as shown in fig.2.2.



Datasets	**Training Data**	**Testing Data**	**Variables**
Prod of Rice in India	687	339	21
Choice of Brand for Crackers	2210	1082	15
Wages and Education of Young Males	2921	1439	13

Fig.2.2 (Sasikala, Biju and Prashanth, 2017)

This study shows the accuracy of algorithm can vary based on the dataset used. The first dataset *(fig.2.3)* shows all the algorithms to be fairly accurate with the lowest accuracy being 0.8942 for KNN. The most variables were used for this dataset, so it confirms the conclusion that more variables lead to a higher accuracy. However, this dataset used the least amount of test data, which could account for the higher accuracy results compared to the other datasets.

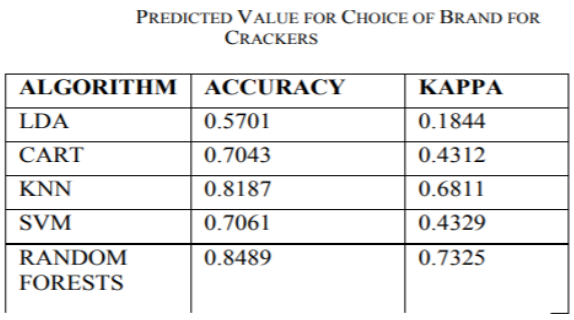


PREDICTED VALUE FOR PRODUCTION OF RICE IN INDIA DATASET

ALGORITHM	**ACCURACY**	**KAPPA**
LDA	0.9135	0.8143
CART	0.9223	0.8753
KNN	0.8942	0.7696
SVM	0.9327	0.8522
RANDOM FORESTS	0.9423	0.8743

Fig.2.3 (Sasikala, Biju and Prashanth, 2017)

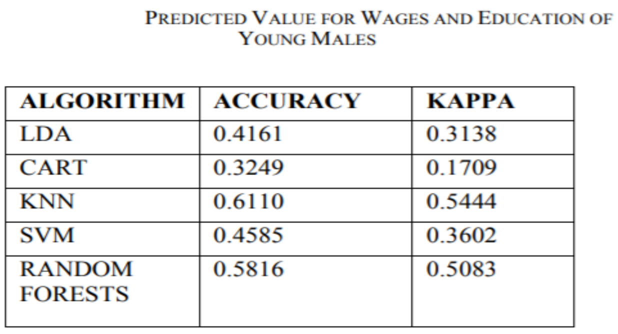
The datasets used in fig.2.4 and fig.2.5 show a lower accuracy for all algorithms, this is most likely due to the lower number of variables. This could also be because there was more testing data used. No conclusion can be drawn from these findings on which algorithms are the most accurate, as the results are inconsistent between datasets.



PREDICTED VALUE FOR CHOICE OF BRAND FOR CRACKERS

ALGORITHM	ACCURACY	KAPPA
LDA	0.5701	0.1844
CART	0.7043	0.4312
KNN	0.8187	0.6811
SVM	0.7061	0.4329
RANDOM FORESTS	0.8489	0.7325

Fig.2.4 (Sasikala, Biju and Prashanth, 2017)

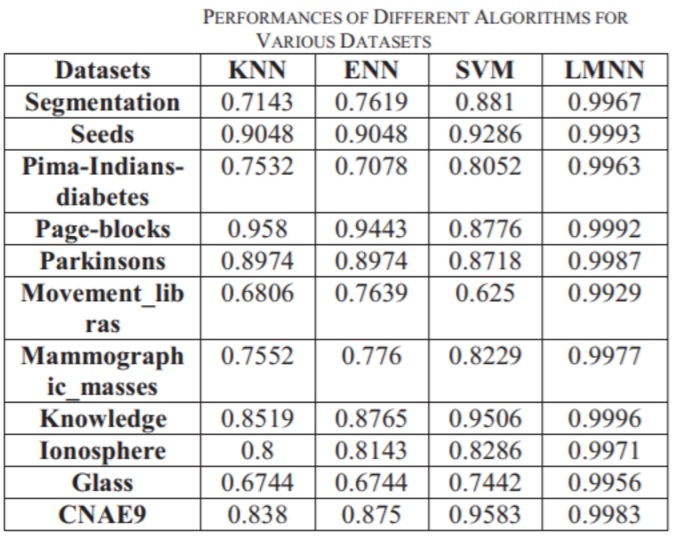


PREDICTED VALUE FOR WAGES AND EDUCATION OF YOUNG MALES

ALGORITHM	ACCURACY	KAPPA
LDA	0.4161	0.3138
CART	0.3249	0.1709
KNN	0.6110	0.5444
SVM	0.4585	0.3602
RANDOM FORESTS	0.5816	0.5083

Fig.2.5 (Sasikala, Biju and Prashanth, 2017)

Another study *(Khan, Arif, Siddique and Oishe, 2018)* showed more reliable results in relation to which algorithms were the most accurate. This study used 11 datasets which all showed similar results which allows more conclusions to be drawn from the data. The findings *(fig.2.6)* show LMNN to be the most accurate algorithm for every dataset with close to 100% accuracy. On average SVM was the second most accurate followed by ENN and KNN with similar accuracy to each other. In conclusion, all of these algorithms are accurate enough for simple problems but if a high degree of accuracy is needed LMNN is the better option although it may be more complex and harder to implement.



PERFORMANCES OF DIFFERENT ALGORITHMS FOR VARIOUS DATASETS

Datasets	KNN	ENN	SVM	LMNN
Segmentation	0.7143	0.7619	0.881	0.9967
Seeds	0.9048	0.9048	0.9286	0.9993
Pima-Indians-diabetes	0.7532	0.7078	0.8052	0.9963
Page-blocks	0.958	0.9443	0.8776	0.9992
Parkinsons	0.8974	0.8974	0.8718	0.9987
Movement_libras	0.6806	0.7639	0.625	0.9929
Mammographic_masses	0.7552	0.776	0.8229	0.9977
Knowledge	0.8519	0.8765	0.9506	0.9996
Ionosphere	0.8	0.8143	0.8286	0.9971
Glass	0.6744	0.6744	0.7442	0.9956
CNAE9	0.838	0.875	0.9583	0.9983

Fig.2.6 (Khan, Arif, Siddique and Oishe, 2017)

2.3 artificial neural networks

Artificial neural networks are algorithms designed after the human brain to recognise patterns. A neural network can be considered as heuristic, meaning they work in a complex non-linear way such as a human brain would. A neuron is a processing unit which connects to other neurons to form a neural network. The memory is stored in the synaptic connections between the neurons rather than the neurons themselves. It is hard to mimic a human brain as it has around 10 to 100 billion neurons connected to many others by trillions of synapses. *(Bapu Ahire, 2018)*

Artificial neural networks consist of an input and output layer as well as one or more hidden layers. The input layers receive data, which can be anything from image and audio data to text but are read in as numbers. The input layers send information via the synaptic connections to the neurons in the hidden layers where the data is processed. The final output is then transmitted to the output layer. The synaptic connections are said to have "weight" which is a number between -1 and 1 in most cases. The weights can change as the neural network is trained; this is known as plasticity. *(Buckley, 2020)*

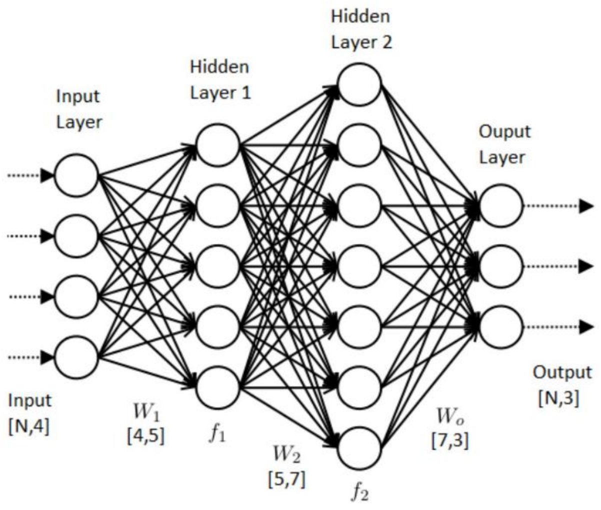


fig.2.7 (Bapu Ahire, 2018)

To train an artificial neural network, the synapses are given a weight to start with, then the numbers from the input layer are multiplied by the weight of the synapses associated with them to then enter the first (or only) hidden layer. Each neuron in the hidden layer sums all the numbers passed into it from the input and inputs them into a function then outputs the result to the synapses to go to the next hidden layer (if there is one). The next hidden layer will repeat the process until the final result is outputted to the output layer.

Once the artificial neural network has a result it can then be compared to the value that it should be equal to. If the value is within a certain threshold of what it should be, the network has been trained successfully, otherwise the weights are adjusted and the training process is repeated. This process will be repeated continuously until the output is within an acceptable threshold of the expected value. Once it is trained, it is expected that the network can find solutions to other problems similar to the one it was trained with.

How good a potential solution is can be represented on a graph known as the search space *(fig.2.8).* An artificial neural network is optimized by navigating the search space to find the best solution it can, however this is not always the best solution possible. There are two types of search space: global search space and cell search space. The global search space represents the entire network and the cell search space represents a certain part of the network.

The search space can be split into segments, the best solution for any segment is known as the local best and the best solution for the whole algorithm is the global best and the worst is the global and local worst (also known as local and global maxima). The local solutions are relative to the neighbouring solutions. There are two ways of searching for maxima and minima, one of these is a grid search. In a grid search, the graph is split into segments where the maxima and minima of each segment is known as the local maxima and minima. There is also a random search, where random x values are generated, and y values calculated from them. These search methods are useful because it allows the search space to be to be narrowed down in order to find the best solutions.

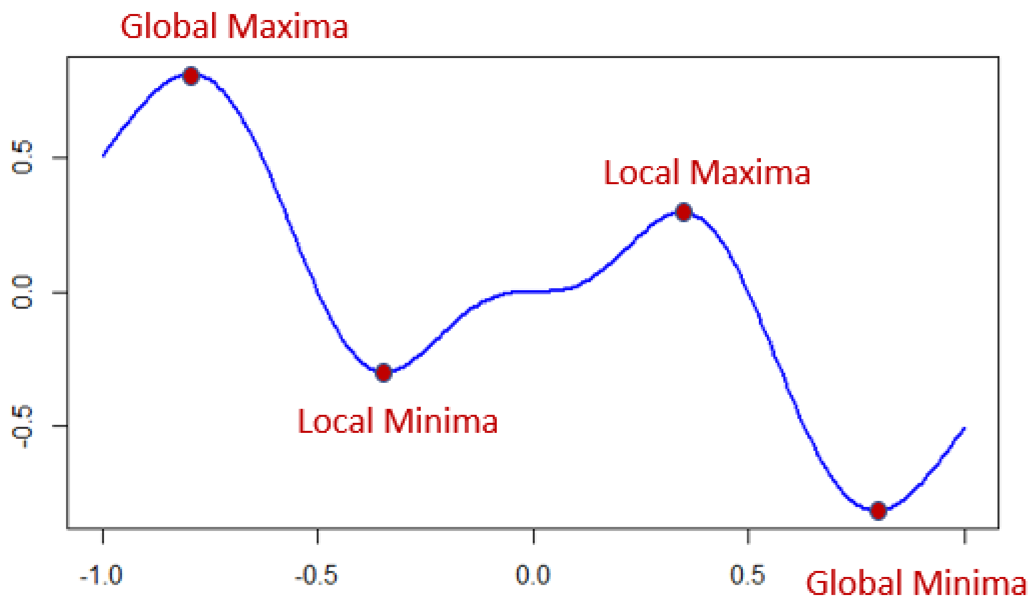


fig.2.8 (Bapu Ahire, 2018)

One study was able to use a neural network to recognise facial expressions *(Kobayashi and Hara 1993).* To achieve this, images of six different facial expressions (surprise, fear, disgust, anger, happiness, and sadness) from three different participants were sampled using a video recorder. The images were recorded at a rate of one image for every 30th of a second in order to show each expression from neutral to maximum, assigning a value of zero to the neutral expression and a value of ten for the maximum expression. The X-Y coordinate system was utilised to map facial features for different facial expressions.

In this study “i” denotes the number of input information, “t” represents the time and “j” the number of output units. A formula is derived from “i” and “t” for an output of “j”. As the neural network was trained, results from the output layers were fed back to the input layer to make the hidden layer for the weights to be adjusted accordingly to improve the accuracy of the neural network. When a root mean squared error of less than 0.01 was achieved, the learning process was terminated.

The graphs *(fig.2.9)* show that the neural network was trained well to dynamically recognise facial expressions. The recognition ratio showed a smooth increase in all graphs as the intensity of the facial expression increased from neutral to maximum. The neural network is not perfectly accurate as it shows a small response for different facial expressions when it should not, for example, when measuring the fear expression, slight spikes appeared in the anger and disgust graphs but not to a significant degree. There are also some unexplained declines, when measuring the fear expression for example. The fear graphs recognition drops when the stimulus number is roughly 5 before increasing again. In conclusion I think this study was accurate enough to dynamically recognise human facial expressions with a high to reasonable degree of accuracy. However, for an outdated study, very impressive degrees of accuracy were shown and with modern resources, the accuracy could be improved even more.

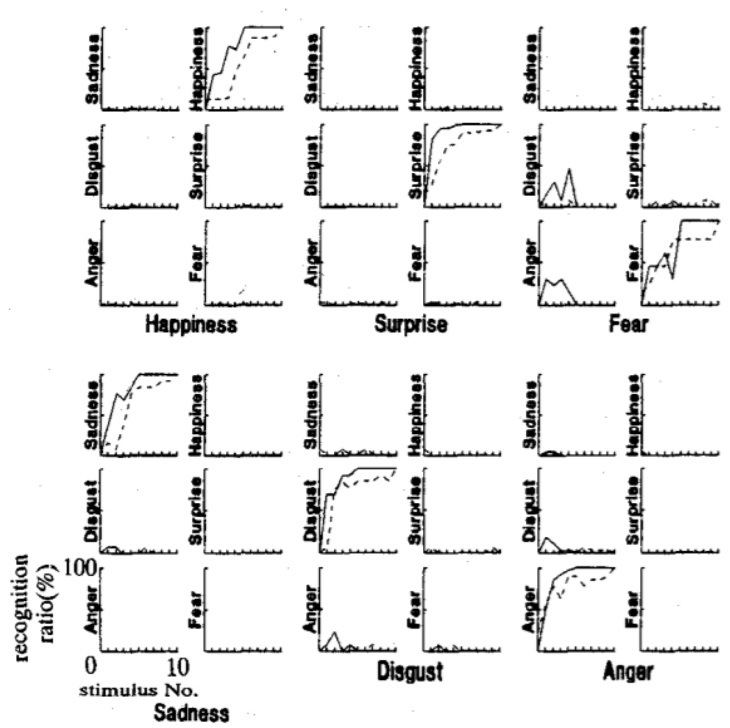


fig.2.9 (Kobayashi and Hara 1993)

2.4 linear regression

Linear regression is a statistical method of modelling how one or more variables (dependant and independent) relate to each other. It can also be used as a machine learning algorithm. In simple linear regression, two variables are used to derive a formula to model the relationship between them, known as the regressor or linear function. The function can be represented on a graph of the two variables as the equation of a straight line that predicts the dependent variable in the most accurate way possible. Given a value of x you can use the function to predict y.

This method can be done by hand by drawing out a graph and drawing a line of best fit. Where a given x value meets the line will be the prediction of the y value. The straight line is described with the equation $y = mx + c$ where x and y are the coordinates, m is the gradient and c is the y intercept. An x value can be substituted into the equation to predict the y value. The line is usually fitted using the least squares method. This method takes the sum of the squared values of the deviation between the line and the points. The final line is the one with the smallest total deviation because a smaller deviation means a more accurate model. The reason the values are squared is to only account for the magnitude of the deviation as it could be negative or positive.

For large datasets or for when a high degree of accuracy is required, the mathematical and hand drawn approaches are less appropriate. Instead, a linear regression algorithm can be used. This is an example of supervised machine learning. Using an algorithm, large datasets can be read in and an accurate function can be derived from them. Using this function, it can almost instantly predict the dependant variable values for any number of independent variable values. The data can also easily be represented on graphs with the linear function represented as a straight line. By using larger datasets, a higher degree of accuracy can be achieved as the algorithm is trained with more data.

This kind of algorithm should only be used for variables that have a near linear relationship in order to keep a high degree of accuracy as it uses a formula based on a straight line.

Multiple variable regression is a more advanced form of linear regression where the dependant variable is predicted based on multiple independent variables. The dependant variable is still predicted using $y = mx + c$ but an adapted version for multiple variables $y = mx_1 + y = mx_2 \dots + mx_n + c + \varepsilon$ where ε is the random error component and n is the number of independent variables used. This method is more suitable for problems where the outcome depends on multiple variables in a linear relationship. An example of where this can be applied is to predict the value of a stock which will depend on multiple factors such as value and momentum.

Fuzzy linear regression is another branch of linear regression which uses the same concepts as statistical linear regression. Similarly, they both assume the relationship between x and y is linear and use this to derive a function. The key difference between the two is usually in linear regression the error is assumed to be random but in fuzzy linear regression, the error is expected due to "fuzziness". With a fuzzy regression model, the dependence between variables is based on multiple fuzzy models. The observed variable values can be considered either real or fuzzy.

A simple way to test the accuracy of an algorithm is to split the data into test and training data (30% test data and 70% training data) and train the algorithm. The test and training data should use values within a similar range because a relationship may not be consistent across all ranges. Once the algorithm has been trained, test data should be used to predict the dependent variables. The value of the predicted dependent variable and the actual dependent variable can then be compared.

The mean absolute error is simply the average difference between the actual value and predicted value. Using the equation $mean\ error = mean(\frac{(actual\ value - predicted\ value)}{actual\ value}) \times 100$ you can find the mean absolute percentage error of the algorithm. The accuracy as a percentage is then calculated by subtracting the error from one hundred. To improve the reliability of the calculated accuracy, multiple data sets should be used with different amounts of variables in each data set. This will determine how accurate an algorithm is as a whole and not just for one dataset. Note that the negative values must be regarded as positive for the error to be considered absolute.

R squared is a value used in statistics that measures how much of the variation in the predicted values is explained by the variation in independent variables. R squared is a value between one and zero with zero meaning the outcome cannot be predicted by the independent variables and one meaning the independent variables can be used to predict without error. This value is calculated using $R^2 = 1 - \frac{sum\ squared\ regression\ error}{sum\ squared\ total\ eror}$ where sum squared regression error represents the unexplained error and sum squared total error representing the total error. Sum squared regression error Is equal to the sum of the errors squared and the sum squared total error is equal to the sum of the difference between each dependant variable and the average dependant variable value squared.

Another value that is commonly used to measure accuracy error is the root mean square and is useful for understanding the magnitude of the deviation of points from the line. The way this is calculated is by squaring the deviations from the line and calculating the square root of the average of these numbers. By squaring the numbers, they are all made positive in order to only observe the magnitude rather than if they are positive or negative. There is no specific range that a root mean squared error should be as it depends on the range of values within the dataset. For example, a dataset with values from 1-1000 should expect a higher root mean squared value than a dataset that ranges from 1-10. Similar to the R squared value, root mean squared error is a measure of the variance of points from the regression line but is an absolute value rather than relative.

One study *(Kavitha, Varuna, and Ramya, 2016)* analysed linear regression and support vector regression. Support vector regression uses either linear or non-linear kernel functions for regression, using a tolerance margin ε. In the study both algorithms were tested to find various measures of the error and accuracy of each. It also accounted for the time to build the algorithms with the linear regression algorithm taking 3.29 seconds to build and the linear kernel model taking 2.42 seconds to build. The difference between both these times is negligible considering the linear regression model had a higher accuracy and less error than the other algorithm on all measurements considered, although the difference in error was not very significant.

In the same study the mean absolute error for the linear regression algorithm was 9.6689 which is 0.4095 less than the other algorithm. This measurement alone would suggest linear regression to be more accurate, but the difference is not too significant and there are better ways to measure accuracy of such algorithms. More conclusions can be drawn from the root relative squared errors which were positive values in both studies but the value for linear regression was 2.6764% less which is quite significant in this case. If both values were of different magnitudes it would be harder to draw a conclusion for root squared errors as they are relative. One of the most reliable measurements of accuracy, root mean squared error, also showed linear regression to be more accurate by a small margin of 0.3365. in conclusion, this study shows the linear regression algorithm to be more accurate than support vector regression for this dataset by a small amount, however it is reasonable to assume that the results could vary using different datasets.

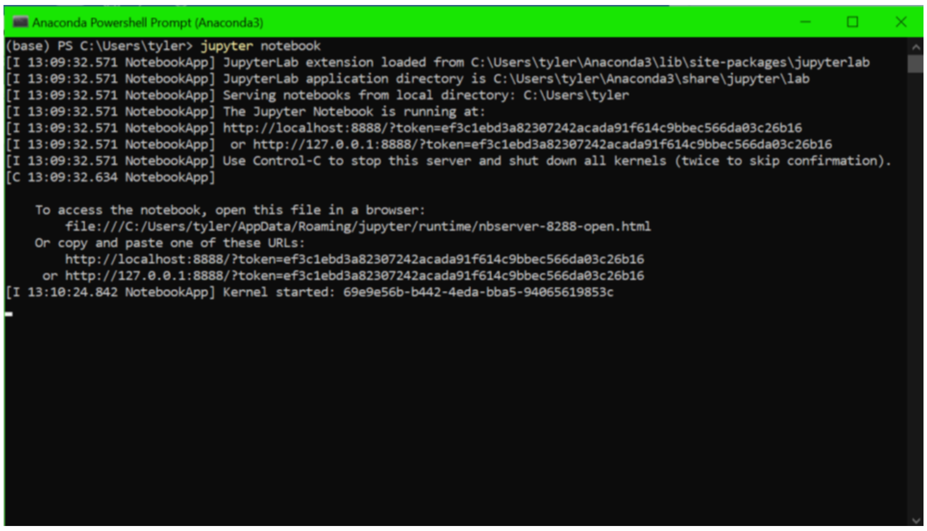
3.0 Methodology and Research Design

3.1 Main code

The goal of the research was to test the linear regression algorithm. To do this the Jupyter notebook feature of the Anaconda software was used (Installation — Anaconda documentation, 2020). This software allows the user to code in Python 3 with access to multiple machine learning libraries. The main benefit to this type of interface is the code is broken down into chunks and it can be run in steps to isolate errors to a chunk of code. The aim was to test the algorithm with a dataset containing basketball statistics *(Ratto, 2020).*

The first step was to find a dataset with two features that had a close to linear relationship between them in order to make the results accurate. Once the dataset was identified, the irrelevant columns were deleted, using Microsoft Excel, and only the columns titled “FG” (field goals made) and “FGA” (field goals attempted) were kept because of their linear relationship. The data was then split into 70% training data and 30% test data.

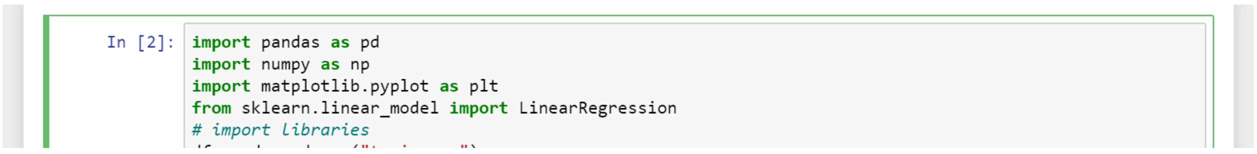
Once Anaconda was installed, the PowerShell prompt was opened and “Jupyter notebook” was entered to access Jupyter notebook and created a new notebook coded in python 3 *(fig.3.1).*



```
(base) PS C:\Users\tyler> jupyter notebook
[I 13:09:32.571 NotebookApp] JupyterLab extension loaded from C:\Users\tyler\Anaconda3\lib\site-packages\jupyterlab
[I 13:09:32.571 NotebookApp] JupyterLab application directory is C:\Users\tyler\Anaconda3\share\jupyter\lab
[I 13:09:32.571 NotebookApp] Serving notebooks from local directory: C:\Users\tyler
[I 13:09:32.571 NotebookApp] The Jupyter Notebook is running at:
[I 13:09:32.571 NotebookApp] http://localhost:8888/?token=ef3c1ebd3a82307242acada91f614c9bbec566da03c26b16
[I 13:09:32.571 NotebookApp]  or http://127.0.0.1:8888/?token=ef3c1ebd3a82307242acada91f614c9bbec566da03c26b16
[I 13:09:32.571 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 13:09:32.634 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///C:/Users/tyler/AppData/Roaming/jupyter/runtime/nbserver-8288-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=ef3c1ebd3a82307242acada91f614c9bbec566da03c26b16
     or http://127.0.0.1:8888/?token=ef3c1ebd3a82307242acada91f614c9bbec566da03c26b16
[I 13:10:24.842 NotebookApp] Kernel started: 69e9e56b-b442-4eda-bba5-94065619853c
```

Fig.3.1

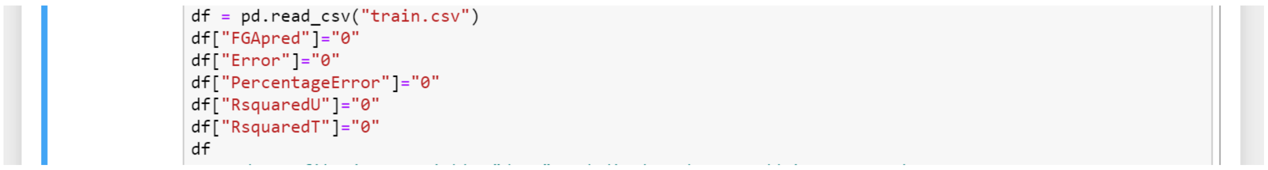
The first lines of code are to import the libraries. The libraries that are needed are: pandas (as pd), NumPy (as np), matplotlib.pyplot (as plt) and Linear Regression from sKlearn.Linear_model. The pandas library is used for manipulating data structures and allows the reading and writing of data to CSV files, which is the format of the dataset used. Pandas also allows the reshaping of data. NumPy allows additional support of large multi-dimensional arrays. Matplotlib.pyplot allows data to be plotted onto a graph and other features related to graphs. Linear regression is used to derive a linear regression formula from an array of data.



```
In [2]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        from sklearn.linear_model import LinearRegression
        # import libraries
```

fig.3.2

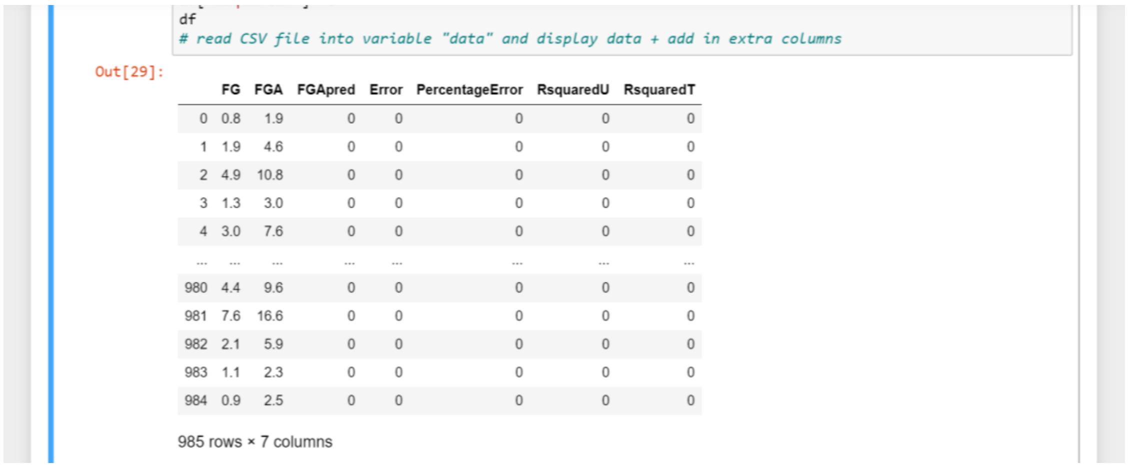
After importing the libraries, the next step is to import the training data and store it in the variable df (data frame) using the code: `df= pd.read_csv("train.csv")`. Next, some other columns were added to the table, which were to be filled later. The first column added was “FGApred” which was for the predicted values of FGA. Also, columns were added for absolute error and absolute percentage error labelled “error” and “percentage error”. The “RsquaredU” and “RsquaredT” columns represent the unexplained variance between actual values and predicted value and also the total variance. On their own the last two columns don’t mean much but they were used to calculate the R squared value further in the code. All of the added columns were set to an initial value of zero.



```
df = pd.read_csv("train.csv")
df["FGApred"]="0"
df["Error"]="0"
df["PercentageError"]="0"
df["RsquaredU"]="0"
df["RsquaredT"]="0"
df
```

fig.3.3

By typing `df` a sample of the data is displayed to ensure it is the correct dataset. The data used has the two original columns FG (field goals made) and FGA (field goals attempted). The data is based on basketball analytics from multiple players over the course of multiple games. It also contains the added fields which at this point all had values of zero.



```
df
# read CSV file into variable "data" and display data + add in extra columns
```

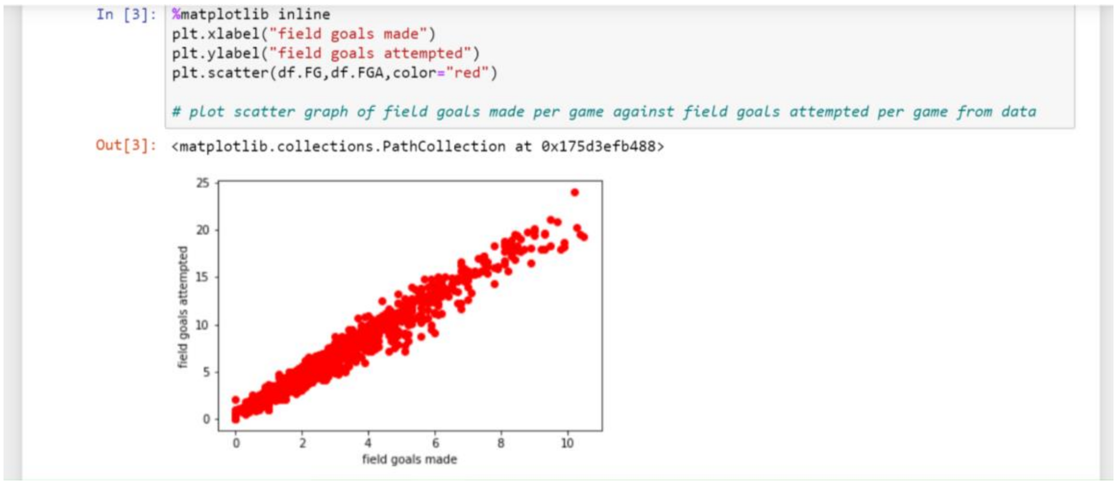
Out[29]:

	FG	FGA	FGApred	Error	PercentageError	RsquaredU	RsquaredT
0	0.8	1.9	0	0	0	0	0
1	1.9	4.6	0	0	0	0	0
2	4.9	10.8	0	0	0	0	0
3	1.3	3.0	0	0	0	0	0
4	3.0	7.6	0	0	0	0	0
...	...	...	...	...	...	...	...
980	4.4	9.6	0	0	0	0	0
981	7.6	16.6	0	0	0	0	0
982	2.1	5.9	0	0	0	0	0
983	1.1	2.3	0	0	0	0	0
984	0.9	2.5	0	0	0	0	0

985 rows × 7 columns

fig.3.4

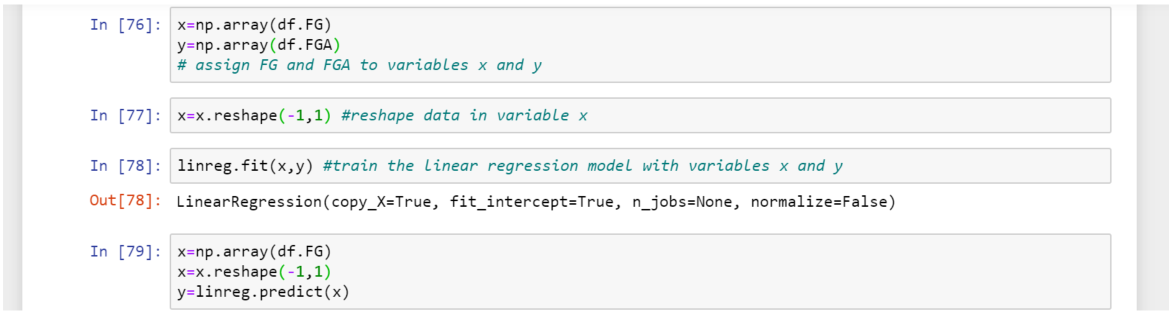
The following code was then used to display the data graphically and plot field goals attempted (Y) against field goals made (X). Then the axis was labelled, and the colour of the plot was changed to red. The graph below shows an almost linear relationship between the variable on each axis which makes it ideal for the linear regression algorithm rather than the other machine learning algorithms.



```
In [3]:  %matplotlib inline
         plt.xlabel("field goals made")
         plt.ylabel("field goals attempted")
         plt.scatter(df.FG,df.FGA,color="red")

         # plot scatter graph of field goals made per game against field goals attempted per game from data

Out[3]:  <matplotlib.collections.PathCollection at 0x175d3efb488>
```

fig.3.5

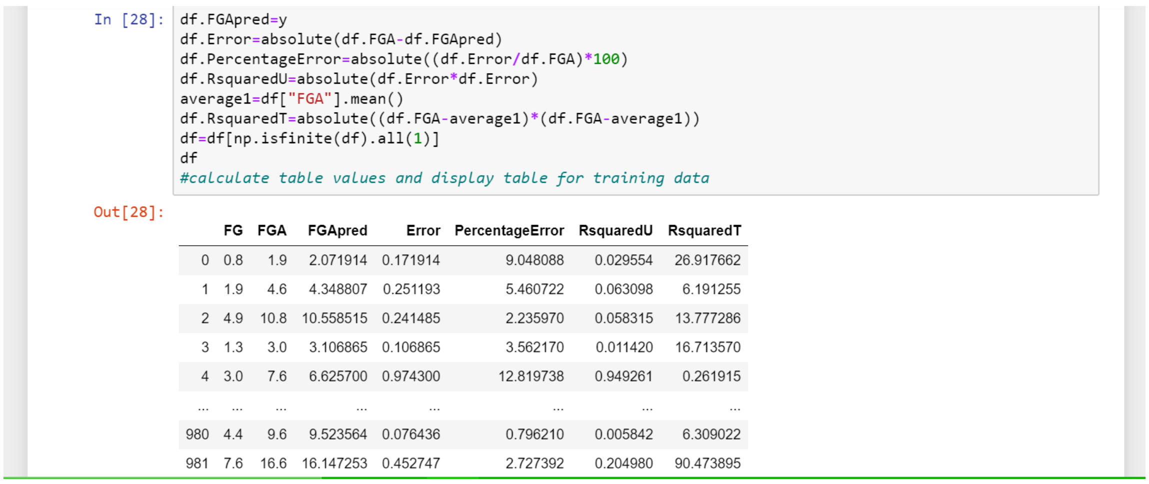
The next step was to assign the array of values in the data frame to the variables "x" for field goals made and "y" for field goals attempted. Also, in order to carry out the next steps, the linear regression function was assigned to the variable "linreg" to create an instance of the linear regression model. The linear regression model expects a two dimensional array so, in order to avoid errors in the next steps, "x" needed to be reshaped using the code `x=x.reshape(-1,1)`. The "-1" keeps all elements in the array the same and the "1" adds the second dimension. After these steps, the linear regression model is ready to be trained with the x and y variables created earlier as shown below. The linear regression model was also used to predict the FGA value for each given FG value to allow the rest of the table to be filled in.



```
In [76]:  x=np.array(df.FG)
          y=np.array(df.FGA)
          # assign FG and FGA to variables x and y

In [77]:  x=x.reshape(-1,1) #reshape data in variable x

In [78]:  linreg.fit(x,y) #train the linear regression model with variables x and y

Out[78]:  LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

In [79]:  x=np.array(df.FG)
          x=x.reshape(-1,1)
          y=linreg.predict(x)
```

fig.3.6

Because x and y were now fitted to the linear regression model and the y values had been predicted, the rest of the table for the training data could be filled in. The previously predicted FGA values (y) were then assigned to the "FGApred" column. The calculations for the other columns were also carried out and assigned to the related columns. These columns were all for analysing the accuracy of how the algorithm was trained. By typing `df` again, a sample of the data was then displayed.

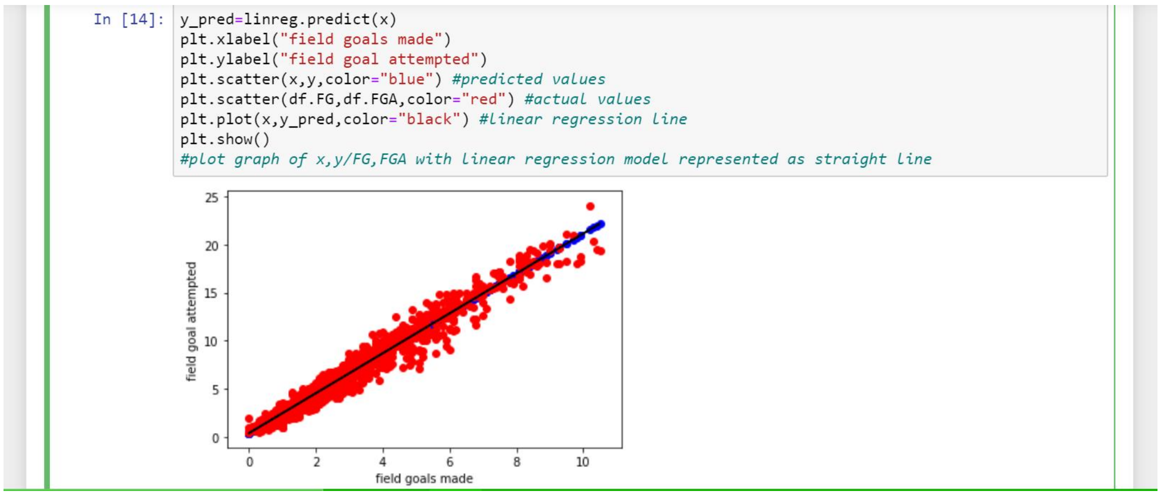


```
In [28]: df.FGApred=y
         df.Error=absolute(df.FGA-df.FGApred)
         df.PercentageError=absolute((df.Error/df.FGA)*100)
         df.RsquaredU=absolute(df.Error*df.Error)
         average1=df["FGA"].mean()
         df.RsquaredT=absolute((df.FGA-average1)*(df.FGA-average1))
         df=df[np.isfinite(df).all(1)]
         df
         #calculate table values and display table for training data

Out[28]:
```

	FG	FGA	FGApred	Error	PercentageError	RsquaredU	RsquaredT
0	0.8	1.9	2.071914	0.171914	9.048088	0.029554	26.917662
1	1.9	4.6	4.348807	0.251193	5.460722	0.063098	6.191255
2	4.9	10.8	10.558515	0.241485	2.235970	0.058315	13.777286
3	1.3	3.0	3.106865	0.106865	3.562170	0.011420	16.713570
4	3.0	7.6	6.625700	0.974300	12.819738	0.949261	0.261915
...	...	...	...	...	...	...	...
980	4.4	9.6	9.523564	0.076436	0.796210	0.005842	6.309022
981	7.6	16.6	16.147253	0.452747	2.727392	0.204980	90.473895

fig.3.7

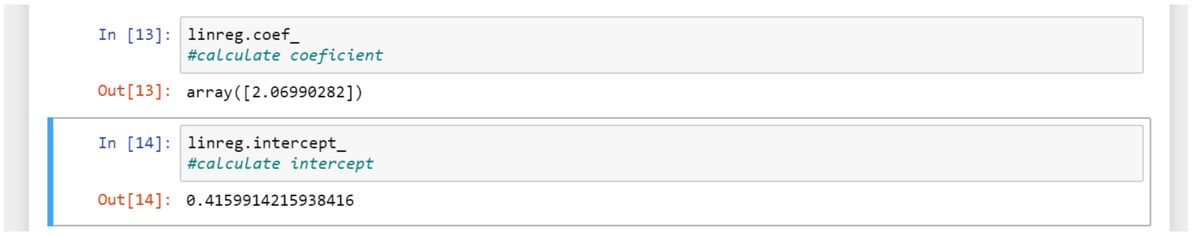
For a visual representation of the algorithm, the original graph was then replotted with this instance of the linear regression model represented as a straight line. Then the colour of the line was changed to black and the colour of the plot to red to differentiate between the two. Also, a plot of FG against predicted FGA in blue for a comparison between the predicted and actual values was added. As shown below they follow the same general linear line, but the predicted results follow the linear regression line perfectly.



```
In [14]: y_pred=linreg.predict(x)
         plt.xlabel("field goals made")
         plt.ylabel("field goal attempted")
         plt.scatter(x,y,color="blue") #predicted values
         plt.scatter(df.FG,df.FGA,color="red") #actual values
         plt.plot(x,y_pred,color="black") #linear regression line
         plt.show()
         #plot graph of x,y/FG,FGA with linear regression model represented as straight line
```

fig.3.8

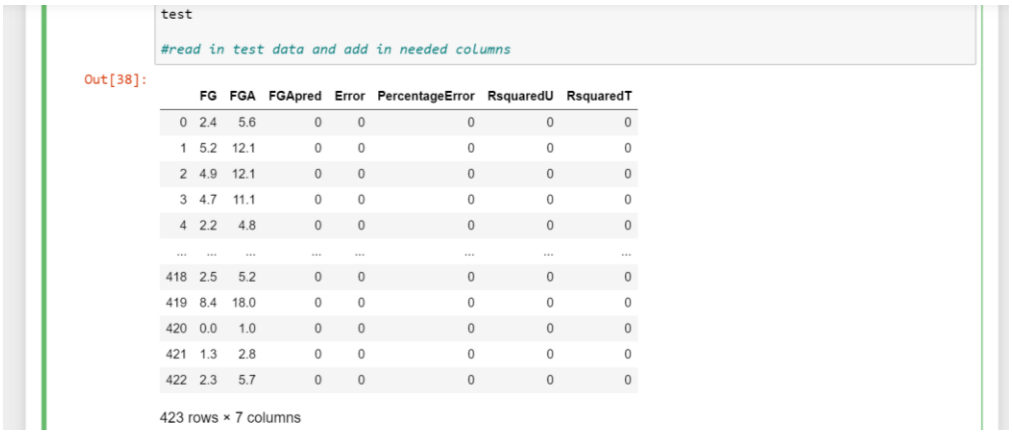
The y intercept and the coefficient were derived. This information can be useful for mathematically testing the results using the formula y=mx+c or to draw the graph out by hand. The intercept was calculated to be 0.4159914215938416 and the coefficient is 2.06990282. Any y value can be predicted for any given x value using these numbers.



```
In [13]: linreg.coef_
         #calculate coeficient

Out[13]: array([2.06990282])

In [14]: linreg.intercept_
         #calculate intercept

Out[14]: 0.4159914215938416
```

fig.3.9

After the algorithm was trained, the test data was read in with the same columns as the training data added in. Below you can see a sample of the test data before the added columns had been filled in.



```
test

#read in test data and add in needed columns
```

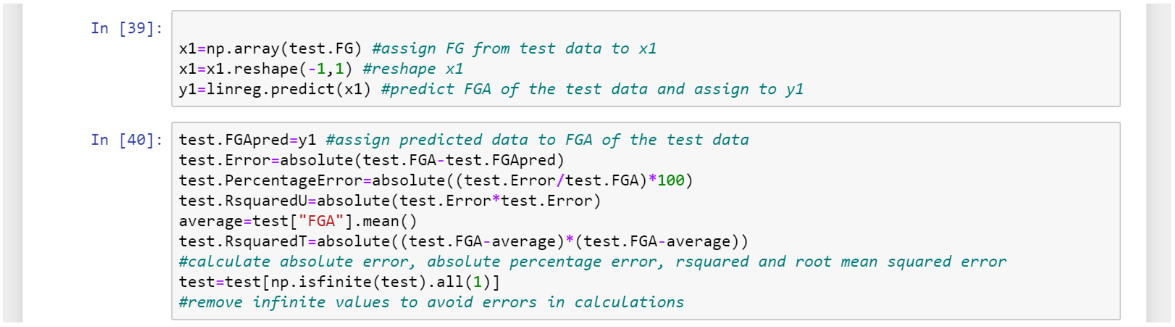
Out[38]:

	FG	FGA	FGApred	Error	PercentageError	RsquaredU	RsquaredT
0	2.4	5.6	0	0	0	0	0
1	5.2	12.1	0	0	0	0	0
2	4.9	12.1	0	0	0	0	0
3	4.7	11.1	0	0	0	0	0
4	2.2	4.8	0	0	0	0	0
...	...	...	...	...	...	...	...
418	2.5	5.2	0	0	0	0	0
419	8.4	18.0	0	0	0	0	0
420	0.0	1.0	0	0	0	0	0
421	1.3	2.8	0	0	0	0	0
422	2.3	5.7	0	0	0	0	0

423 rows × 7 columns

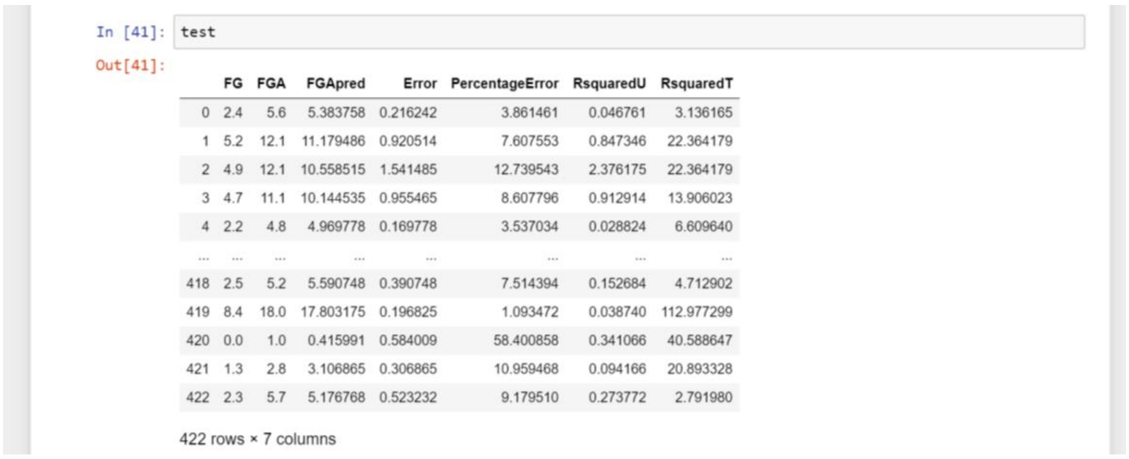
fig.3.10

The next step was to assign the FG column of the test data to x1 and then reshape the array to two dimensions to avoid errors. To predict the next column, `y1=linreg.predict(x1)` was used. This uses the same instance of the algorithm that was trained earlier but with the FG (x1) values used to predict the corresponding FGA (y1) from the test data rather than the training data. Now the values were predicted, they could then be assigned to the "FGApred" column allowing the rest of the table to be filled in. note the line `test=test[np.isfinite(test).all(1)]` was to remove infinite values from the table that were affecting the errors calculated. After this line of code one of the rows was removed from the test data. These errors were most likely caused by the data containing values of NaN, infinity, or a value too large. Then displayed was a sample of the data `test`.



```
In [39]: x1=np.array(test.FG) #assign FG from test data to x1
         x1=x1.reshape(-1,1) #reshape x1
         y1=linreg.predict(x1) #predict FGA of the test data and assign to y1

In [40]: test.FGApred=y1 #assign predicted data to FGA of the test data
         test.Error=absolute(test.FGA-test.FGApred)
         test.PercentageError=absolute((test.Error/test.FGA)*100)
         test.RsquaredU=absolute(test.Error*test.Error)
         average=test["FGA"].mean()
         test.RsquaredT=absolute((test.FGA-average)*(test.FGA-average))
         #calculate absolute error, absolute percentage error, rsquared and root mean squared error
         test=test[np.isfinite(test).all(1)]
         #remove infinite values to avoid errors in calculations
```

fig.3.11



```
In [41]: test
Out[41]:
```

	FG	FGA	FGApred	Error	PercentageError	RsquaredU	RsquaredT
0	2.4	5.6	5.383758	0.216242	3.861461	0.046761	3.136165
1	5.2	12.1	11.179486	0.920514	7.607553	0.847346	22.364179
2	4.9	12.1	10.558515	1.541485	12.739543	2.376175	22.364179
3	4.7	11.1	10.144535	0.955465	8.607796	0.912914	13.906023
4	2.2	4.8	4.969778	0.169778	3.537034	0.028824	6.609640
...	...	...	...	...	...	...	...
418	2.5	5.2	5.590748	0.390748	7.514394	0.152684	4.712902
419	8.4	18.0	17.803175	0.196825	1.093472	0.038740	112.977299
420	0.0	1.0	0.415991	0.584009	58.400858	0.341066	40.588647
421	1.3	2.8	3.106865	0.306865	10.959468	0.094166	20.893328
422	2.3	5.7	5.176768	0.523232	9.179510	0.273772	2.791980

422 rows × 7 columns

fig.3.12

For a visual representation the data was plotted on a graph *(fig.3.13)*. The predicted points all lie on the black line from the previous graph (coloured blue on this graph), which represented the linear regression model. This shows that the results were predicted based on this line. Also displayed in red are the points plotted from the original data.

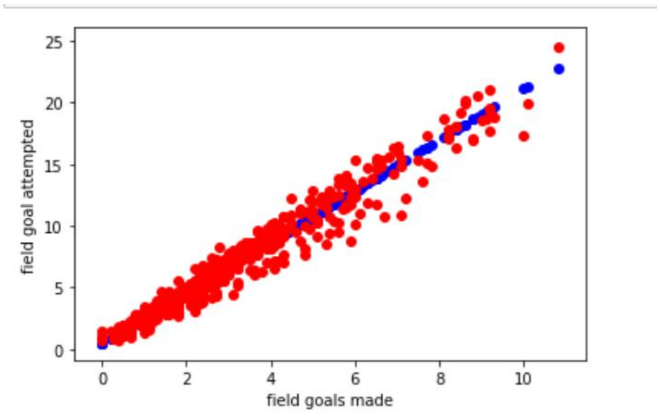
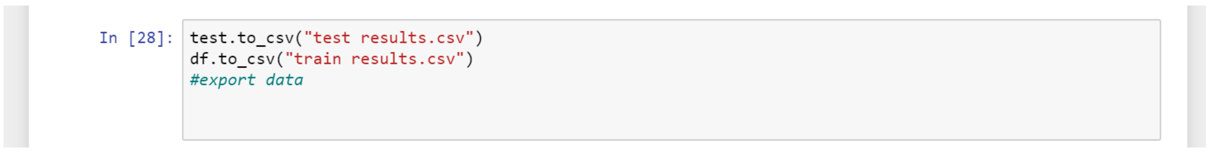


Fig.3.13

Then the final results from the test and training data that included all the columns for errors were exported to CSV files using the following code.



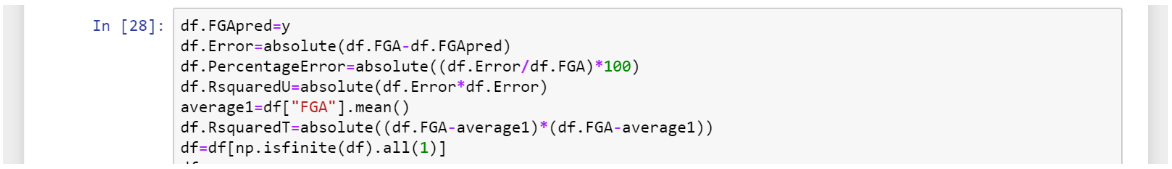
```
In [28]: test.to_csv("test results.csv")
         df.to_csv("train results.csv")
         #export data
```

fig.3.14

3.2 Errors

To calculate the absolute errors of my results the equation $predicted\ value - original\ value$ was used then then the equation $\frac{error}{original\ value} \times 100$ to calculate the percentage error. To calculate a value representing the unexplained variance (RsquaredU column) the error was simply squared and to represent the total variance (RsquaredT) the equation $(original\ value - average)^2$ was used.

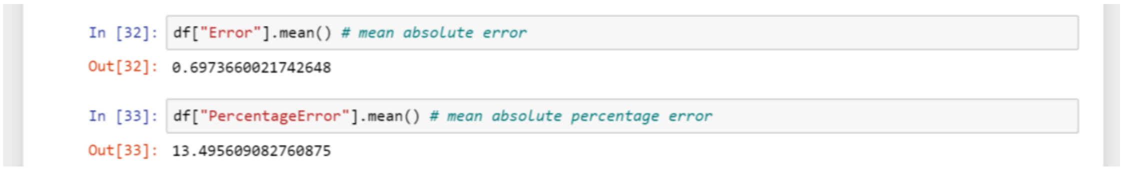
Average represents the average original value of the full dataset. The last two calculations don't mean much on their own but are useful for calculating the R squared and root mean squared error. All values used for these equations were the absolute values, this means only the magnitude is used regardless if the number is negative or positive, however this is made simple by the absolute function from NumPy. Below is a screenshot of how this is coded based on the dataset used in order to fill in the full columns.



```
In [28]: df.FGApred=y
         df.Error=absolute(df.FGA-df.FGApred)
         df.PercentageError=absolute((df.Error/df.FGA)*100)
         df.RsquaredU=absolute(df.Error*df.Error)
         average1=df["FGA"].mean()
         df.RsquaredT=absolute((df.FGA-average1)*(df.FGA-average1))
         df=df[np.isfinite(df).all(1)]
```

fig.3.15

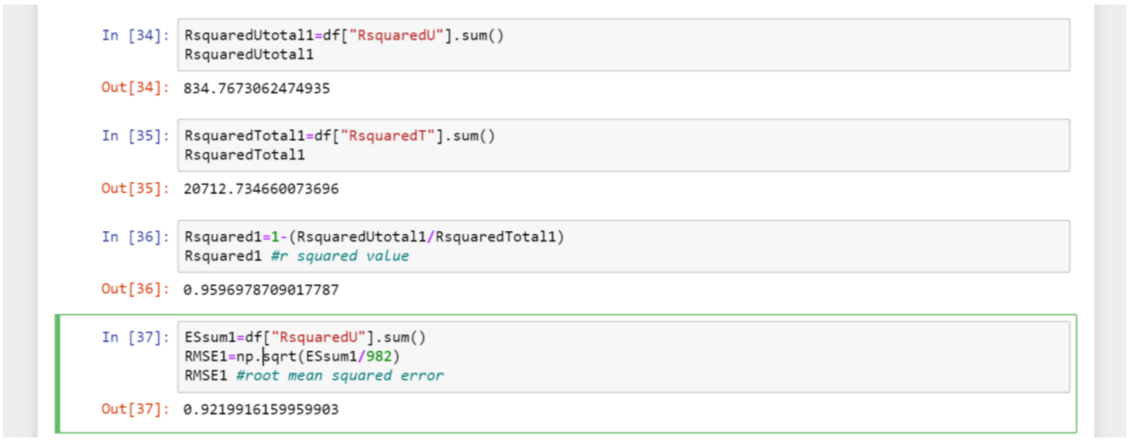
With all these columns filled in the mean absolute error, mean percentage error, R squared, and root mean squared errors can be easily calculated. The mean absolute error and mean percentage error are calculated by simply finding the average value from the corresponding columns created earlier. This can be done using the `mean()` function as shown below.



```
In [32]: df["Error"].mean() # mean absolute error

Out[32]: 0.6973660021742648

In [33]: df["PercentageError"].mean() # mean absolute percentage error

Out[33]: 13.495609082760875
```

fig.3.16

To calculate the R squared value, I summed all the "RsquaredU" and "RsquaredT" using the `sum()` function. Then the equation $1 - \frac{RsquaredU(sum)}{RsquaredT(sum)}$ was used to calculate the R squared value for the full dataset. The root mean squared error was then calculated by taking the sum of the "RsquaredU" values and then dividing it by the total number of "RsquaredU" values. The final result is then square rooted to calculate the root mean squared error of the full dataset. For this calculation the `sum()` function was used again along with the square root function from NumPy `np.sqrt`. All of the error calculations detailed above were used for the training data and then repeated on the test data.



```
In [34]: RsquaredUtotal1=df["RsquaredU"].sum()
         RsquaredUtotal1

Out[34]: 834.7673062474935

In [35]: RsquaredTotal1=df["RsquaredT"].sum()
         RsquaredTotal1

Out[35]: 20712.734660073696

In [36]: Rsquared1=1-(RsquaredUtotal1/RsquaredTotal1)
         Rsquared1 #r squared value

Out[36]: 0.9596978709017787

In [37]: ESsum1=df["RsquaredU"].sum()
         RMSE1=np.sqrt(ESsum1/982)
         RMSE1 #root mean squared error

Out[37]: 0.9219916159959903
```

fig.3.17

4.0 Results and Discussion

The mean absolute error for the training data of the algorithm used was 0.697 with an absolute percentage error of 13.496%, which although not that significant of an error, could have marginally affected how accurately the algorithm was trained. Similar to the training data, the test data had an absolute error of 0.762 and absolute percentage error of 13.040% which would suggest it can generalise reasonably on its learning. However, this is not the best measure of accuracy for an algorithm such as this, more commonly used measures of accuracy are R squared and root mean squared. The reason why the absolute errors don't mean much is because one anomaly can significantly change these values.

The training data achieved an R squared value of 0.960 which is very accurate considering R squared values are on a scale of zero to one with one being perfectly accurate. R squares represents how close the raw data points are to the regression line. The value achieved suggests that the training data had a strong linear relationship between both variables and was appropriate to be trained with this algorithm. The test data's R squared value was 0.950 which is almost the same as for the training data, suggesting it generalised on its learning almost perfectly. This also shows that the test and training values follow a very similar trend. Based on only the R squared value, the testing and training of the algorithm with the dataset used was successful. Note absolute values were used to calculate the R squared value in order to draw more conclusions from the value.

Root mean squared error is also a measure of how closely the points fit the regression line but unlike R squared, it is an absolute value rather than relative. The data trained had a root mean squared error of 0.922, which means the predicted data was less than a value of one off the original data. As the Y axis was plotted on a scale of 0-25, this value is good but could be improved, however for the purpose of sports analytics, this value is acceptable. A root mean squared error of below one is generally accepted as adequate, but it can depend on the range of the values used in the dataset.

Both the test *(fig.3.13)* and training data *(fig.3.8)* were plotted. These plots show both the raw data plots in red and the plot where the y values have been predicted. On both the plots the blue points were plotted very close to the red points, showing the algorithm is realistic and predicts values close to the real values. The first plot supports the idea that the algorithm trained well because the red points were so close to the regression line. The algorithm also generalised well on its training as shown by the red points on the second plot also following the same pattern as the blue points and the regression line.

Another study that recorded the accuracy of the linear regression algorithm *(Kavitha, Varuna and Ramya, 2016)* was only able to achieve a mean absolute error of 9.6689 and a root mean squared error of 12.54 for a total of 1000 instances *(fig.4.1)*. These errors are much higher than the ones in this research despite using less data. The main reason why the errors are so much higher is likely because it was trained with less appropriate data that does not follow the same linear trends. Another reason for the higher errors could be because the values used in the data were higher so as a fraction of the highest values, the error would not account for much. In comparison to the study *(Kavitha, Varuna and Ramya, 2016)*, the research carried out above has very little error. The root squared errors in this study suggest it is more accurate than the other measurements, however the study used relative values, so it is hard to draw a comparison between this and a study using absolute values.

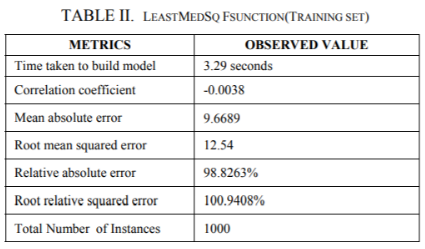


TABLE II. LEASTMEDSQ FSUNCTION(TRAINING SET)

METRICS	OBSERVED VALUE
Time taken to build model	3.29 seconds
Correlation coefficient	-0.0038
Mean absolute error	9.6689
Root mean squared error	12.54
Relative absolute error	98.8263%
Root relative squared error	100.9408%
Total Number of Instances	1000

fig.4.1 (Kavitha, Varuna and Ramya, 2016)

5.0 Conclusion

In conclusion the linear regression algorithm is accurate in most cases and will be adequate for any data that follows linear, or close to linear, trends. Other machine learning algorithms are also useful for different types of data, for example clustering is useful where the data needs to be split into groups. For even more accurate results, it might be necessary to combine algorithms together or use other more complex methods of each algorithm such as multivariable linear regression. All machine learning algorithms have their uses from stock market analysis to sporting analytics.

The root mean squared error and r squared are both good methods of measuring the accuracy of predictions. Absolute errors and percentage errors have their place also but are not the best way of measuring errors under all circumstances. The algorithm trained had high accuracy based on all these error calculations proving it to be effective for certain problems. Linear regression is not appropriate for datasets that do not follow a strong linear trend between variables, instead other algorithms should be considered.

In the future research should be carried out to make more accurate linear regression models that can generalise on their learning to be used with multiple datasets. Also, more research should be done on multivariable linear regression because the more variables taken into account, the more accurate the algorithm should be. Therefore, if a highly efficient algorithm that could take large amounts of variables into consideration was to be developed, new levels of accuracy could be achieved. Other machine learning algorithms should also be explored and developed further, or even new algorithms developed, so with a combination of multiple evolved algorithms, data will be predicted in a more efficient and accurate way than ever before. Finally, more research should be done on ways to improve efficiency of these algorithms so large datasets can be processed quickly. The uses of machine learning are endless, and it is likely many more uses will be discovered in the future.

6.0 Bibliography

- Computinghistory.org.uk. (2020). Arthur Lee Samuel - Computing History. [online] Available at: http://www.computinghistory.org.uk/det/6455/Arthur-Lee-Samuel/ [Accessed 30 Jan. 2020].
- Bolen, A. (2020). Online Fraud. [online] Sas.com. Available at: https://www.sas.com/en_gb/insights/articles/analytics/can-advanced-analytics-for-credit-scoring-change-the-mortgage-market.html [Accessed 2 Feb. 2020].
- Pahwa, K. and Agarwal, N. (2019). Stock Market Analysis using Supervised Machine Learning. [conference paper] IEEE. Available at: https://ieeexplore.ieee.org/document/8862225 [Accessed 3 Feb. 2020].
- Sasikala, B., Biju, V. and Prashanth, C. (2017). Kappa and accuracy evaluations of machine learning classifiers. [conference paper] IEEE. Available at: https://ieeexplore.ieee.org/document/8256551 [Accessed 5 Feb. 2020].
- Khan, M., Arif, R., Siddique, M. and Oishe, M. (2017). Study and Observation of the Variation of Accuracies of KNN, SVM, LMNN, ENN Algorithms on Eleven Different Datasets from UCI Machine Learning Repository. [conference paper] IEEE. Available at: https://ieeexplore.ieee.org/document/8628041 [Accessed 13 Feb. 2020].
- Docs.anaconda.com. 2020. Installation — Anaconda Documentation. [online] Available at: <https://docs.anaconda.com/anaconda/install/> [Accessed 14 March 2020].
- Ratto, D., 2020. NBA PLAYERS 2016-2019. [online] Kaggle.com. Available at: <https://www.kaggle.com/davra98/nba-players-20162019> [Accessed 14 March 2020].
- Hideo, H., Yusuke, S. and Kei, H. (2012). NNRMLR: A Combined Method of Nearest Neighbor Regression and Multiple Linear Regression. [conference paper] IEEE. Available at: https://ieeexplore.ieee.org/document/6337221 [Accessed 6 April. 2020].
- H. Shakouri, G. R. Nadimi, and F. Ghaderi (2012). Fuzzy linear regression models with absolute errors and optimum uncertainty. [conference paper] IEEE. Available at: https://ieeexplore.ieee.org/document/4419325 [Accessed 6 April. 2020].
- Wu, H., Okutani, I., and Xu, J. (2009). Identification of Band Reflectance and Land-cover Classification Using Fuzzy Linear Regression Analysis. [conference paper] IEEE. Available at: https://ieeexplore.ieee.org/document/5358913 [Accessed 6 April. 2020].
- Tai, S., Kuo, T., and Li, K. (2013) An Efficient Super Resolution Algorithm Using Simple Linear Regression. [conference paper] IEEE. Available at: https://ieeexplore.ieee.org/document/6830031 [Accessed 13 April. 2020].
- Kavitha, S., Varuna, S., and Ramya R. (2016). A comparative analysis on linear regression and support vector regression [conference paper] IEEE. Available at: https://ieeexplore.ieee.org/document/7916627 [Accessed 13 April. 2020].
- Bishop, C., (2016). Pattern Recognition And Machine Learning. 1st ed. New York: Springer.
- Buckley, N., (2020). Artificial Neural Networks [presentation] Moodle. Available at: https://live.moodle.hope.ac.uk/course/view.php?id=231 [Accessed 17 April. 2020].
- Bapu Ahire, J., (2018). The Artificial Neural Networks Handbook: Part 1. [online] Medium. Available at: <https://medium.com/coinmonks/the-artificial-neural-networks-handbook-part-1-f9ceb0e376b4> [Accessed 18 April 2020].
- Kobayashi, H., Hara, F., (1993). Dynamic recognition of basic facial expressions by discrete-time recurrent neural network [conference paper] IEEE. Available at: https://ieeexplore.ieee.org/document/713882 [Accessed 22 April 2020].

7.0 Appendices

7.1 Appendix 1- full code input

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from numpy import absolute, mean
df = pd.read_csv("train.csv")
df["FGApred"]="0"
df["Error"]="0"
df["PercentageError"]="0"
df["RsquaredU"]="0"
df["RsquaredT"]="0"
df

%matplotlib inline
plt.xlabel("field goals made")
plt.ylabel("field goals attempted")
plt.scatter(df.FG,df.FGA,color="red")

x=np.array(df.FG)
y=np.array(df.FGA)

linreg=LinearRegression()
x=x.reshape(-1,1)

linreg.fit(x,y)

x=np.array(df.FG)
x=x.reshape(-1,1)
y=linreg.predict(x)
```

```
df.FGApred=y
df.Error=absolute(df.FGA-df.FGApred)
df.PercentageError=absolute((df.Error/df.FGA)*100)
df.RsquaredU=absolute(df.Error*df.Error)
average1=df["FGA"].mean()
df.RsquaredT=absolute((df.FGA-average1)*(df.FGA-average1))
df=df[np.isfinite(df).all(1)]
df

y_pred=linreg.predict(x)
plt.xlabel("field goals made")
plt.ylabel("field goal attempted")
plt.scatter(x,y,color="blue")
plt.scatter(df.FG,df.FGA,color="red")
plt.plot(x,y_pred,color="black")
plt.show()

linreg.coef_

linreg.intercept_

df["Error"].mean()

df["PercentageError"].mean()

RsquaredUtotal1=df["RsquaredU"].sum()
RsquaredUtotal1

RsquaredTotal1=df["RsquaredT"].sum()
RsquaredTotal1
```

```
Rsquared1=1-(RsquaredUtotal1/RsquaredTotal1)

Rsquared1


ESsum1=df["RsquaredU"].sum()

RMSE1=np.sqrt(ESsum1/982)

RMSE1


test=pd.read_csv("test.csv")

test["FGApred"]="0"

test["Error"]="0"

test["PercentageError"]="0"

test["RsquaredU"]="0"

test["RsquaredT"]="0"

test


x1=np.array(test.FG)

x1=x1.reshape(-1,1)

y1=linreg.predict(x1)


test.FGApred=y1

test.Error=absolute(test.FGA-test.FGApred)

test.PercentageError=absolute((test.Error/test.FGA)*100)

test.RsquaredU=absolute(test.Error*test.Error)

average=test["FGA"].mean()

test.RsquaredT=absolute((test.FGA-average)*(test.FGA-average))

test=test[np.isfinite(test).all(1)]


test


test["Error"].mean()


test["PercentageError"].mean()
```

```
RsquaredUtotal=test["RsquaredU"].sum()
RsquaredUtotal

RsquaredTtotal=test["RsquaredT"].sum()
RsquaredTtotal

Rsquared=1-(RsquaredUtotal/RsquaredTtotal)
Rsquared

ESsum=test["RsquaredU"].sum()
RMSE=np.sqrt(ESsum/422)
RMSE

plt.xlabel("field goals made")
plt.ylabel("field goal attempted")
plt.scatter(x1,y1,color="blue")
plt.scatter(test.FG,test.FGA,color="red")
plt.show()

test.to_csv("test results.csv")
df.to_csv("train results")
```

7.2 appendix 2- raw test data sample

FG	FGA
2.4	5.6
5.2	12.1
4.9	12.1
4.7	11.1
2.2	4.8
2.7	7.3
1.2	3.3
1.9	4.4
2.6	4.2

7.3 appendix 3- raw training data sample

FG	FGA
0.8	1.9
1.9	4.6
4.9	10.8
1.3	3
3	7.6
5.6	11.8
3.6	7.1
1	2.7
2.9	5.7

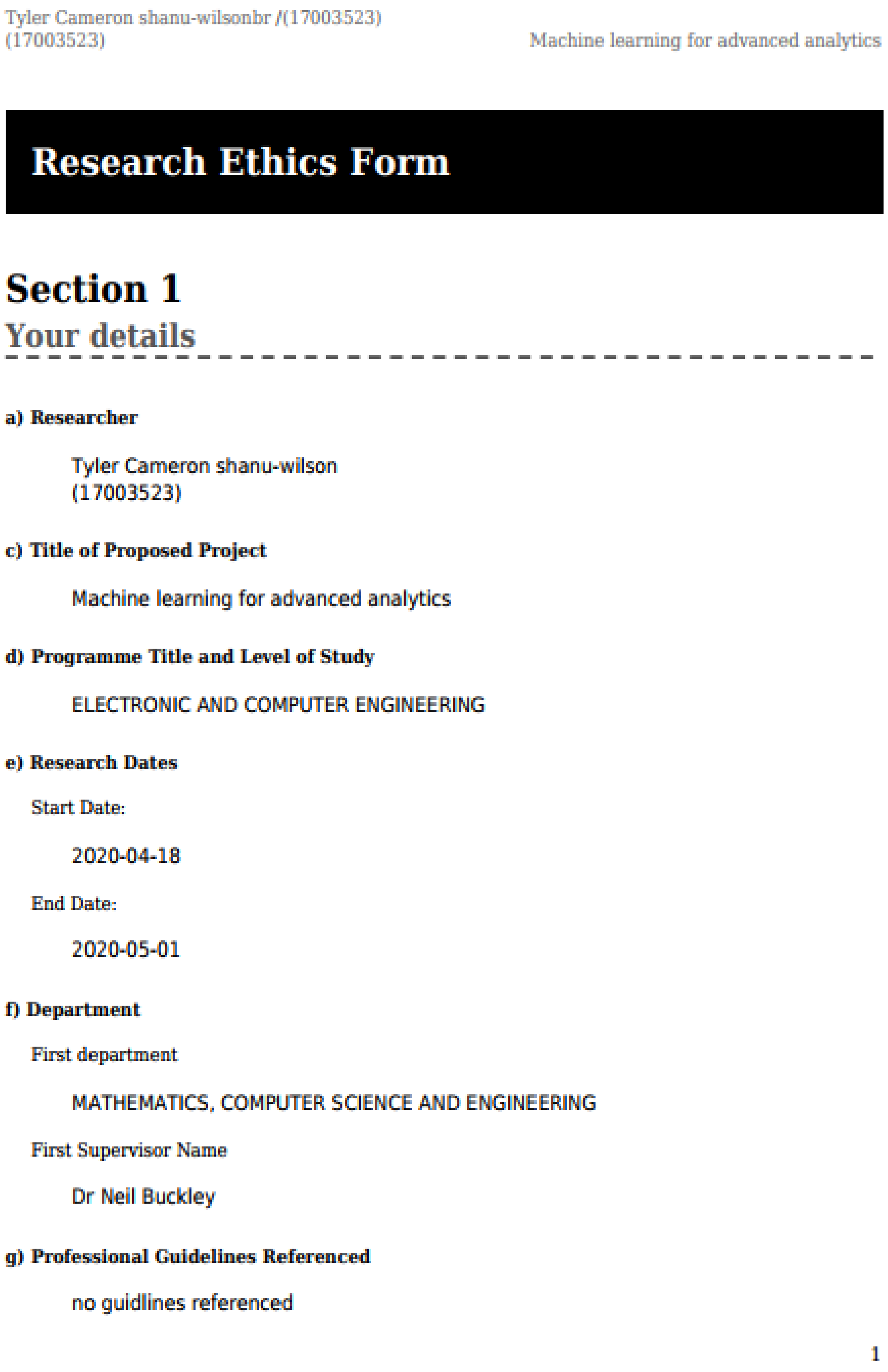
7.4 appendix 4- test data results sample

	FG	FGA	FGApred	Error	PercentageError	RsquaredU	RsquaredT
0	2.4	5.6	5.383758	0.216242	3.861461	0.046761	3.136165
1	5.2	12.1	11.17949	0.920514	7.607553	0.847346	22.36418
2	4.9	12.1	10.55852	1.541485	12.73954	2.376175	22.36418
3	4.7	11.1	10.14453	0.955465	8.607796	0.912914	13.90602
4	2.2	4.8	4.969778	0.169778	3.537034	0.028824	6.60964
5	2.7	7.3	6.004729	1.295271	17.74344	1.677727	0.00503
6	1.2	3.3	2.899875	0.400125	12.12501	0.1601	16.57241
7	1.9	4.4	4.348807	0.051193	1.163482	0.002621	8.826377
8	2.6	4.2	5.797739	1.597739	38.0414	2.552769	10.05475

7.5 appendix 5- training data results sample

	FG	FGA	FGApred	Error	PercentageError	RsquaredU	RsquaredT
0	0.8	1.9	2.071914	0.171914	9.048088	0.029554	26.91766
1	1.9	4.6	4.348807	0.251193	5.460722	0.063098	6.191255
2	4.9	10.8	10.55852	0.241485	2.23597	0.058315	13.77729
3	1.3	3	3.106865	0.106865	3.56217	0.01142	16.71357
4	3	7.6	6.6257	0.9743	12.81974	0.949261	0.261915
5	5.6	11.8	12.00745	0.207447	1.758027	0.043034	22.20084
6	3.6	7.1	7.867642	0.767642	10.81185	0.589274	0.000139
7	1	2.7	2.485894	0.214106	7.929843	0.045841	19.2565
8	2.9	5.7	6.41871	0.71871	12.60894	0.516543	1.927164

7.6 appendix 6- research ethics form



Tyler Cameron shanu-wilsonbr /(17003523) (17003523) Machine learning for advanced analytics

Research Ethics Form

Section 1

Your details

a) Researcher

Tyler Cameron shanu-wilson
(17003523)

c) Title of Proposed Project

Machine learning for advanced analytics

d) Programme Title and Level of Study

ELECTRONIC AND COMPUTER ENGINEERING

e) Research Dates

Start Date:

2020-04-18

End Date:

2020-05-01

f) Department

First department

MATHEMATICS, COMPUTER SCIENCE AND ENGINEERING

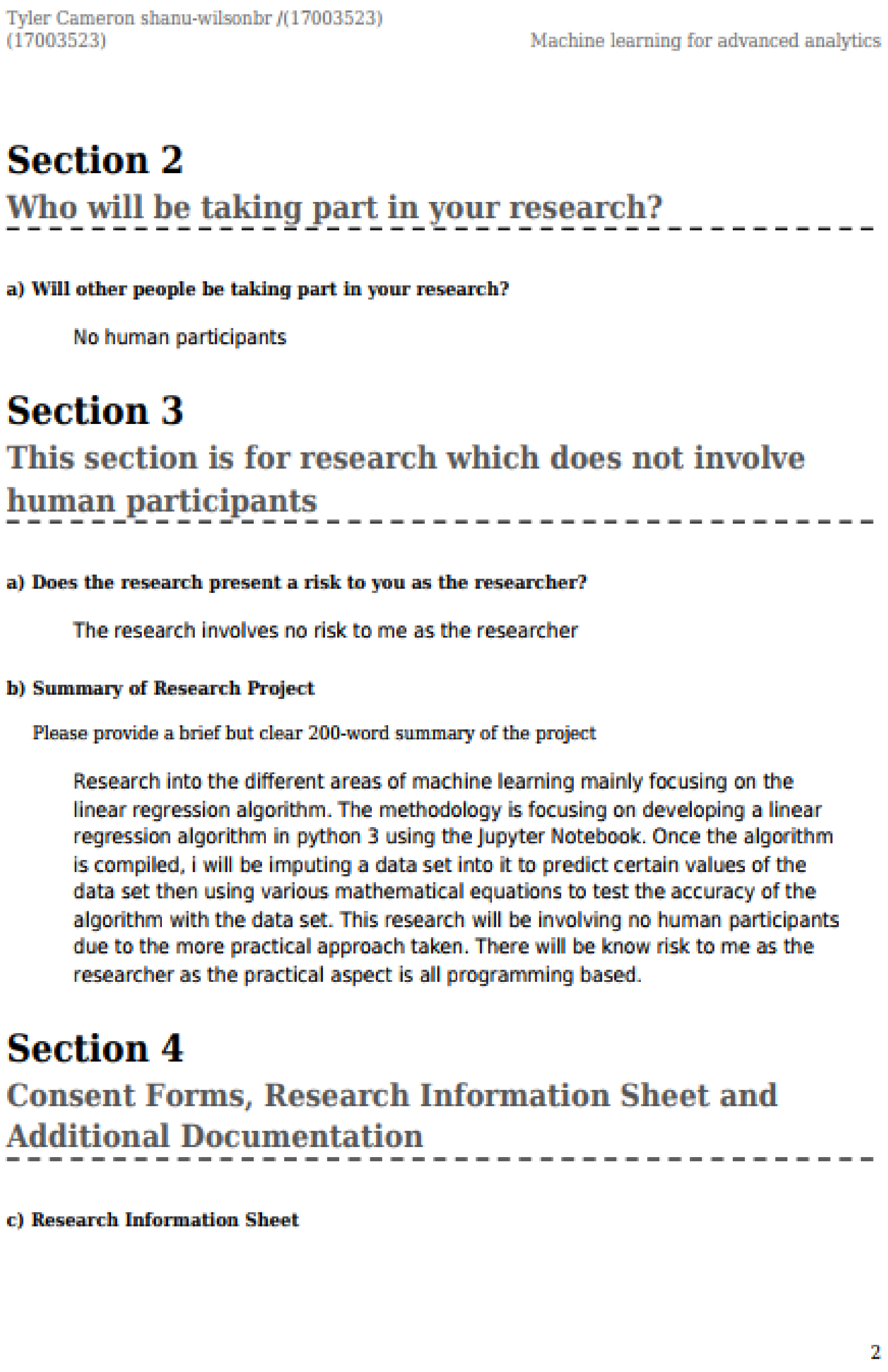
First Supervisor Name

Dr Neil Buckley

g) Professional Guidelines Referenced

no guidlines referenced

1



Section 2

Who will be taking part in your research?

a) Will other people be taking part in your research?

No human participants

Section 3

This section is for research which does not involve human participants

a) Does the research present a risk to you as the researcher?

The research involves no risk to me as the researcher

b) Summary of Research Project

Please provide a brief but clear 200-word summary of the project

Research into the different areas of machine learning mainly focusing on the linear regression algorithm. The methodology is focusing on developing a linear regression algorithm in python 3 using the Jupyter Notebook. Once the algorithm is compiled, i will be imputing a data set into it to predict certain values of the data set then using various mathematical equations to test the accuracy of the algorithm with the data set. This research will be involving no human participants due to the more practical approach taken. There will be know risk to me as the researcher as the practical aspect is all programming based.

Section 4

Consent Forms, Research Information Sheet and Additional Documentation

c) Research Information Sheet

Uploaded Files:

No Files Uploaded

d) Additional Documentation

Uploaded Files:

No Files Uploaded