****

**Assignment Cover Sheet**

To be completed **electronically** by the student and submitted with each piece of work. Please upload this completed cover sheet via Turnitin

**Assignment Title:** Experiment Portfolio – 40%

**Tutor:** Usman Ahmad

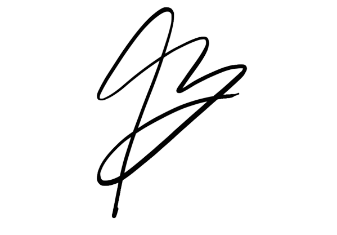
**Student Name: Adhenz Miranda**

**Student Number: 521536**

**Date of Submission:**

**Details of your submission**

In submitting this assignment, I am confirming that I have read and understood the regulations for assessment, and I am aware of the seriousness with which the University regards unfair practice. Please see the universities [Unfair Practice Policy](https://thehub.bathspa.ac.uk/reference/student-policies/unfair-practice-policy) for details.

Signed: Date:

**Title of Experiment 1: House Prices Prediction**

Making the House Prices Prediction was one of the first experiments I was able to do, firstly I imported the modules which are Pandas, Seaborn, and Sci-kit Libraries. Then I got the following dataset on house prices, from that I loaded and read the data to know and understand and input the appropriate values. Which leads me to pre-processing the data, since there is already a determined axis that the data that should be plotted, from X1 to X6 goes on the x variable and the Y goes on the y variable. From there I split the data by using the train\_test\_split() function. Then it leads into predicting the data, the model I used here in this case is the Linear Regression Algorithm as it was deemed appropriate for this dataset and we’re predicting how House Prices could go up over time. The accuracy of this model contains the following:

|  |  |
| --- | --- |
| Mean Squared Error | 53.51 |
| Root Mean Squared | 7.31 |
| R-Squared\*100 | 68.11% |

While arguably not the best accuracy, this is also a prediction model with a continuous value and understand how close it is supposed to be linearly, so in a way, compared to how it actually was, it was not far off to how it was expected to be predicted that house prices actually went up over time.

|  |  |
| --- | --- |
| Kaggle Link |  |

**Title of Experiment 2: Mobile Prices Prediction**

Creating the Mobile Prices Prediction is quite peculiar, because despite the title, it is in fact not similar to house prices, if anything, it was more of a classification problem because in the dataset it was about price range and none of values that had an independent variable were continuous like the house prices, there were about 19 dependent variables and I had to map them into their appropriate variable. The model that I used is KNeighbors Classification function. The accuracy with the use of the following model is:

|  |  |
| --- | --- |
| Accuracy Score | 94.25% |

I used different models like the Decision Tree and Random Forest, but this particular model had the highest accuracy out of all. I assumed it would be quite low like the House Prices Prediction, since it was a classification problem, it is quite accurate. I was initially confused at first on what if it was a regression type prediction I should do or a classification, but after some proper digging of information and looking back at the course I took on LinkedIn, it classifies as a classification dataset at the end.

|  |  |
| --- | --- |
| Kaggle Link |  |

**Title of Experiment 3: Heart Disease Prediction**

I then moved on to making the Heart Disease Prediction, this particular dataset was pretty different compared to the previous two through it using classification, since most models would need numerical values, this particular dataset contains strings and not just integers or floats which both previous datasets only had. Pre-processing this required mapping these particular strings into their specific numerical values, like gender for example, since their strings were ’F’ and ‘M’, I would have to attribute to ‘0’ and ‘1’ respectively, same thing goes for angina, their strings were ‘N’ and ‘Y’ with ‘0’ and ‘1’. After I had mapped the data, I put them in their respective x and y variables. I then split the data then predicted the data. Since this dataset is of classification of who has heart disease or not, I used the Random Forest Classifier. With this algorithm this is the result of the accuracy using the accuracy\_score() function:

|  |  |
| --- | --- |
| Random Forest Classification Accuracy | 88.02% |

It took quite a few runs and the accuracy fluctuates ~88% to 86%. This is quite decent accuracy in detecting heart disease, visualizing through a confusion matrix, there seems to be a false positive and false negatives, though it is still surprisingly good result in finding out who has heart disease or not. I used different models such as KNeighbors and Decision Tree, but they ended up with quite a low accuracy of ~75% to ~70%. Random Forest had the highest accuracy out of all.

|  |  |
| --- | --- |
| Kaggle Link |  |

**Title of Experiment 4: Drugs Classification**

Which moved on to classifying drugs, surprisingly quite similar to the Heart Disease Prediction, the dataset is also quite similar, I also had to map the dataset, just like the previous dataset, those that had strings, I had to and change it to numerical values, Since it is classification in this dataset, there were multiple drug types, there were 5 types of drugs that needed to be classified, so I had to map those too. After mapping the data in their dedicated numerical values in the pre-processing phase. The model I used was the Decision Tree Classifier. The table shows the following accuracy using the accuracy\_score() function:

|  |  |
| --- | --- |
| Decision Tree Classifier | 99.17& |

It was an astounding 99.17% accuracy when using this model. I originally used a different classification algorithm like Random Forest Classification Model, but I wanted to try to get a higher accuracy which was around 95%, which is already high enough. What I did I chose a different classification model/algorithm which in this case was the Decision Tree Classification algorithm, and that resulted in such a very high accuracy score of 99.17%. Visualized in a confusion matrix heatmap, about 0.03 of that data were off in one cell and all of the following had the value of 1.00, with one cell being the value of 0.97. Almost quite perfect I accuracy I must say.

|  |  |
| --- | --- |
| Kaggle Link |  |

**Title of Experiment 5: Spam or Ham Prediction**

With the spam or ham prediction, this was a bit tricky to map, looking at the dataset, I knew it would be a simple classification dataset, with only one dependent variable which were the messages and one independent variable which were if the message were spam or ham. What bugged me was that mapping the spam and ham was the same as others where you convert the strings into numerical values, I had to figure out how do I map the messages into a numerical value since that is what the models only support is if they are floats and integers. But after tinkering around I figured that the best way was to convert the string length of those characters and make them into an array and using the reshape function, it counts the number of rows the array has and totals it into an integer that the machine is able to read and be trained from. After that I trained the data with train\_test\_split() function. The model I used for Spam Ham Prediction is the Decision Tree Classifier function, the following is the accuracy of the model:

|  |  |
| --- | --- |
| Decision Tree Classifier Accuracy | 87.71% |

I tested with other models like the other classification problems, this model had the highest accuracy of all, while it is not one of the most effective ways of detecting spam or ham from the length of the string of characters, the accuracy of the model is still surprisingly quite high with this model.

|  |  |
| --- | --- |
| Kaggle Link |  |

**Title of Experiment 6: Flowers Classification**

With flowers classification it was quite simple, the dataset has been imported built-in from sklearn by importing the load\_iris dataset, instead of reading the CSV datasets, we just need to make a data frame instead. Pre-processing data, since the independent variable we want is species, we map those integers from 0 to 2 and from that, input the flower species. Afterwards we use the drop function to remove the species column so when it gets trained and just use the remaining column in that data frame to be our dependent variable instead so it is not affected, from that dropped species column, we use that column to be our independent variable which will be the column we dropped which will be the species. The model I used is the KNeighbors Classifier, the following is the accuracy of the model used:

|  |  |
| --- | --- |
| KNeigbors Classifier Accuracy | 100% |

The whole model is able to do it at 100% accuracy which means this particular model is already perfect in its own right and it has been trained properly.

|  |  |
| --- | --- |
| Kaggle Link |  |

**LinkedIn Course Completion Certificate (Optional)**

*//Please paste your LinkedIn course completion certificate.*

*A computer code with a certificate

Description automatically generated with medium confidence*