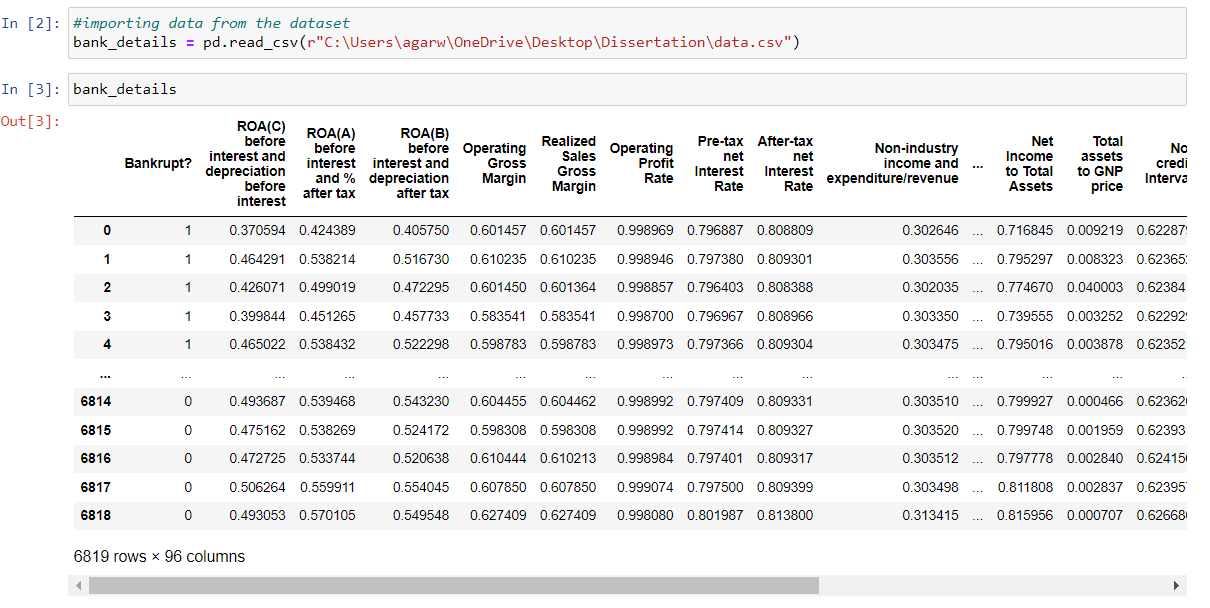
**K-NEAREST NEIGHBORS (KNN) METHOD**

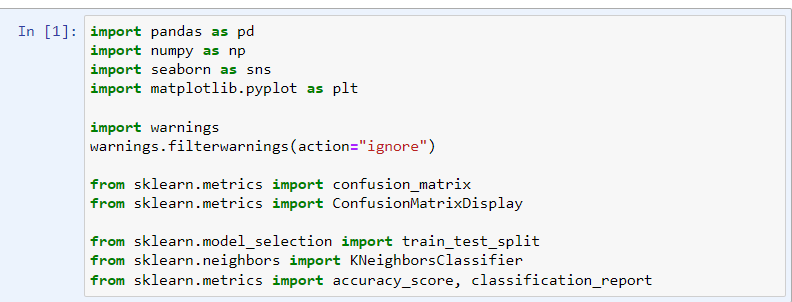
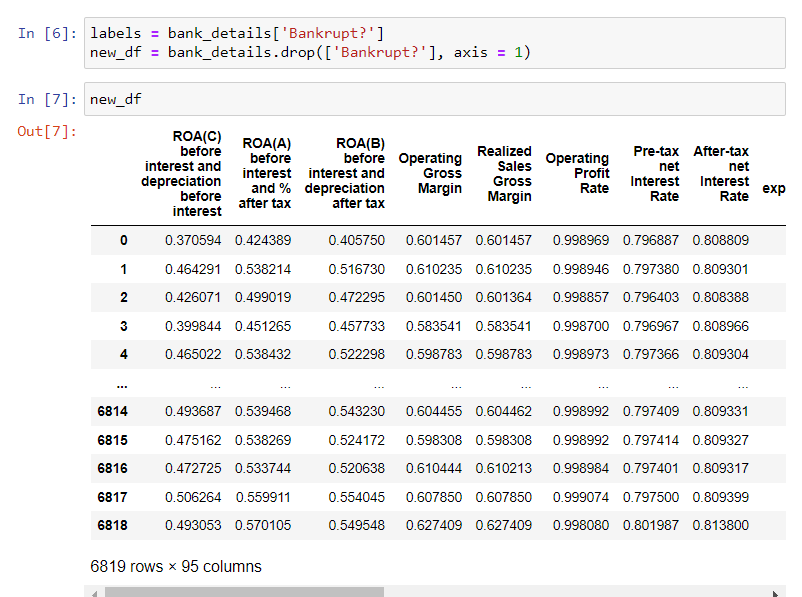
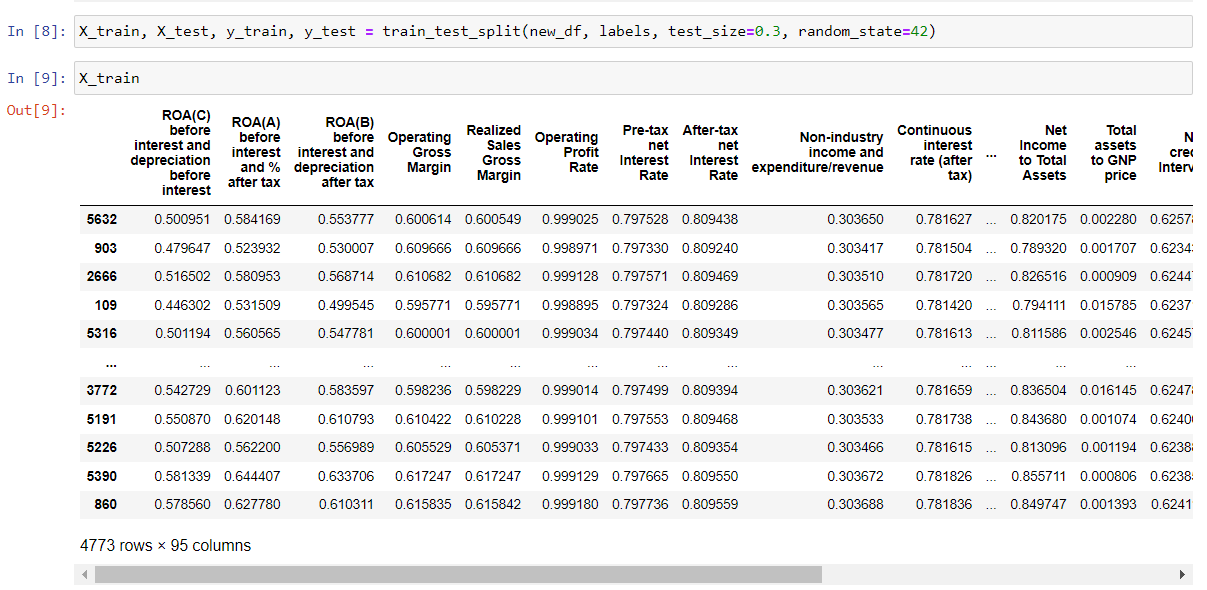
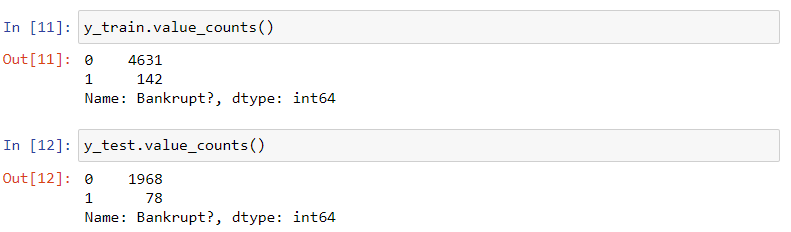
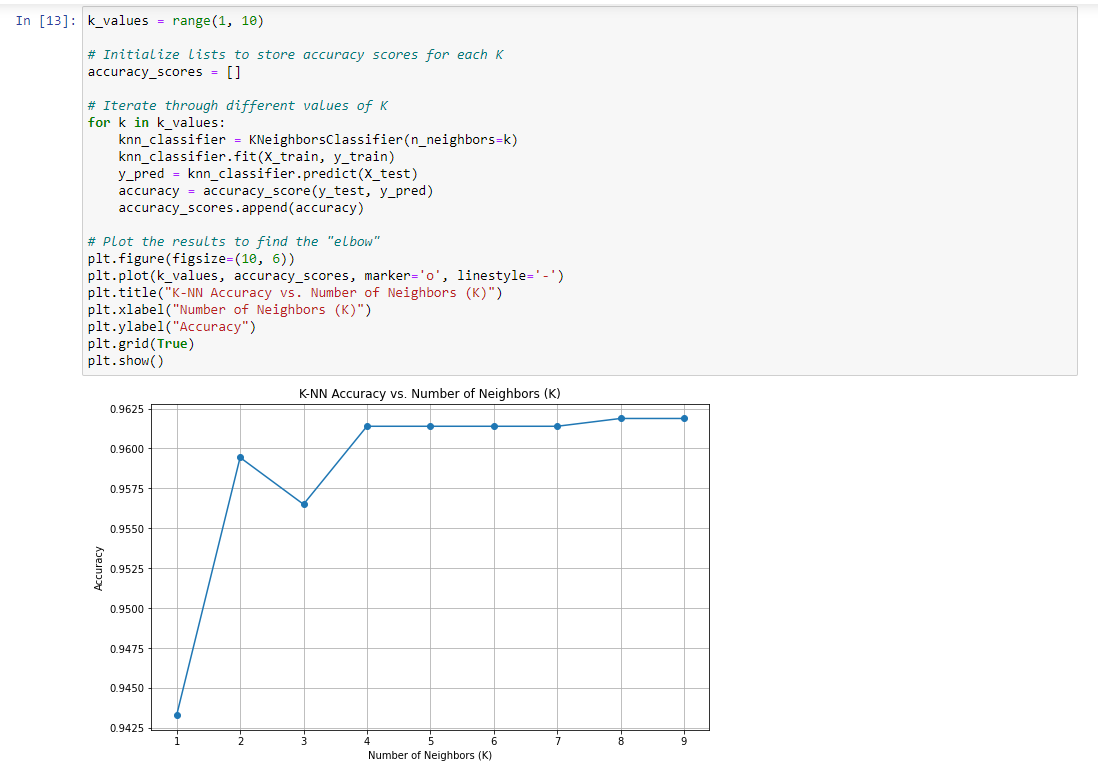
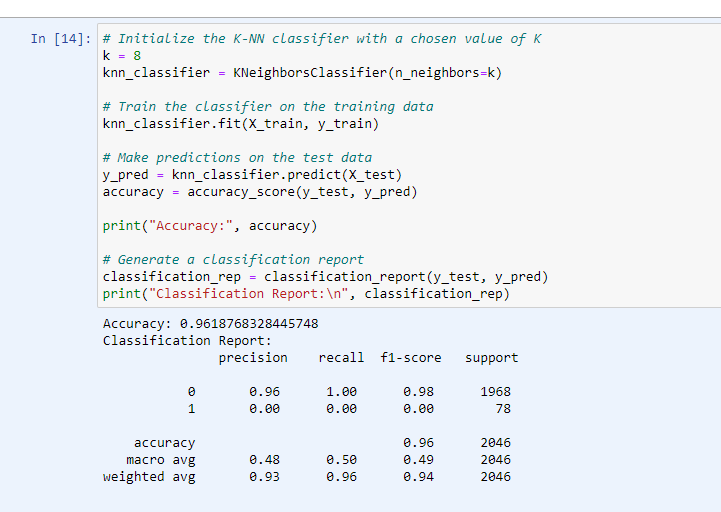
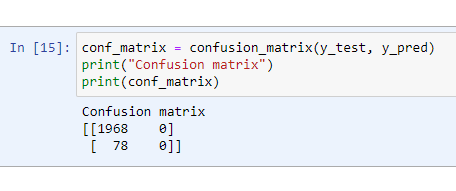
**Darpan Agarwal – K2279745**

**Supervisor - Prof. Nebel, Jean-Christophe A**

**DATASET CONSIDERED**

* I am using Taiwan Companies Financial Dataset for my research.
* The Dataset is publicly available so there is no problem with GDPR laws as it has been extensively researched in the past.
* The Dataset has 6819 records and 96 attributes. 95 are financial indicators and 1 is a label that identifies whether the company is bankrupt or not.
* 
* The Dataset has 6599 companies that are not bankrupt and 220 companies that are bankrupt.
* 

**K-NEAREST NEIGHBORS (KNN) METHOD**

* It is a simple and effective method used for classification tasks.
* In binary classification problems, it can be applied to predict whether a data point belongs to one of the two classes i.e., yes/no, or 1/0, etc. based on the majority class among its K nearest neighbors in the feature space.
* I first imported the required libraries to build our model.
* ****
* Then I rearranged my Dataset by dropping the Bankrupt column and assigning it to the new variable.
* The new dataset had 95 columns instead of 96 in our original.
* ****
* I used the train\_test\_split method to divide my dataset into training and testing set.
* ****
* The parameters we have used are:
  + New\_df is our all columns except for the Bankrupt label.
  + Labels are the Bankrupt columns
  + I have kept the testing dataset at 30% of our total dataset and 70% for our training Dataset.
  + The number of entries for X\_train is 4773 records and X\_test at 2046 records. (4773 + 2046 = 6819)
  + The entries for the bankrupt labels are as follows
  + ****
* After I have applied the train test split method, I can apply the KNN method.
* I needed to find an optimal number of Neighbors to get the best accuracy possible for the dataset.
* For this I have written a range function that loops over the KNN method to find the corresponding accuracy and used the plotting function to show it in the graph.
* ****
* After observing the graph, the accuracy becomes constant after 8.
* Using K = 8 in our KNN method we get the accuracy and classification report.
* ****
* The model has achieved 0.9618 accuracy.
* After getting our accuracy we can apply a confusion matrix for the model.
* ****
* The True Negative (TN) score for our model is 1968 while False Negative (FN) was 78.
* We are not getting any True positive (TP) or False Positive (FP) values as the dataset is highly imbalanced.

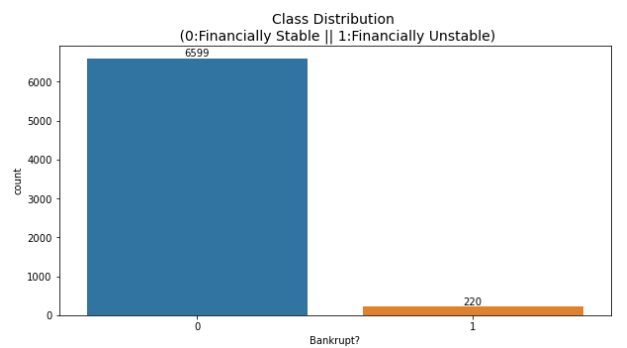
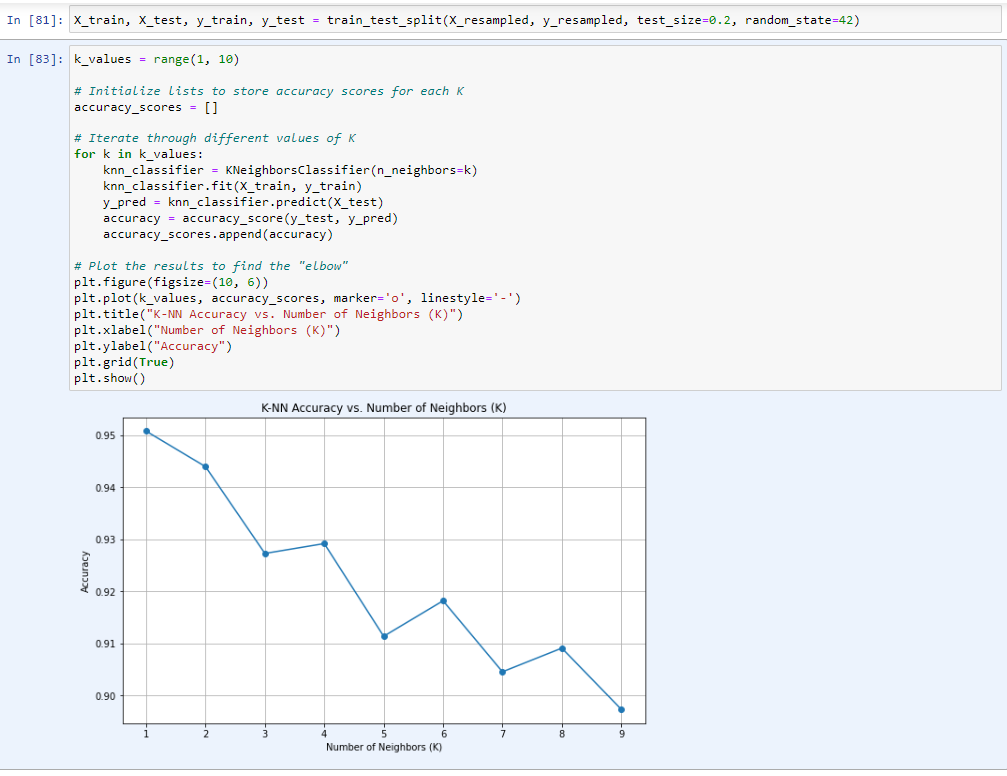
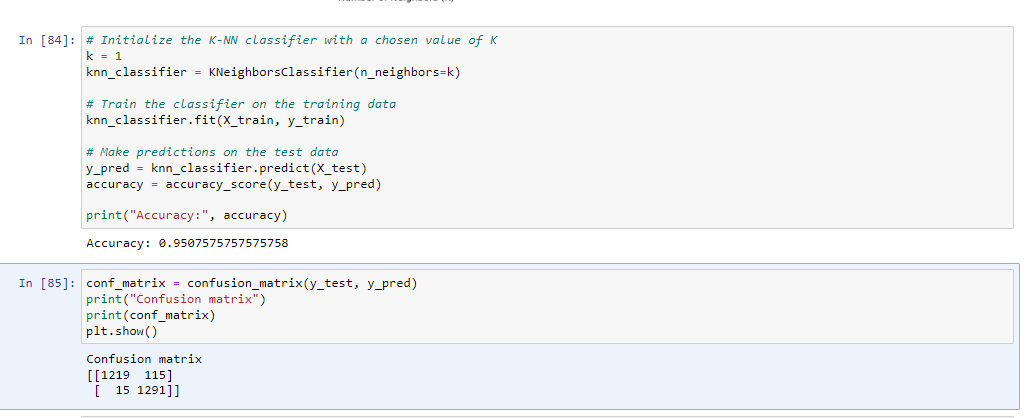
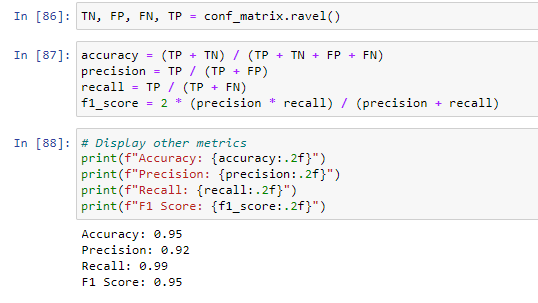
**CONFUSION MATRIX**

* Confusion matrix is a very effective tool to evaluate the performance of classification algorithms, including the K-nearest neighbors (KNN) method.
* It provides a summary of the predictions made by the model compared to the actual class labels in a classification problem.
* It has 4 main keys:
  + True Positive (TP): The number of instances correctly identified as the positive class.
  + True Negative (TN): The number of instances correctly identified as the negative class.
  + False Positive (FP): The number of instances incorrectly classified as the negative class.
  + False Negative (FN): The number of instances incorrectly classified as the negative class that were actually positive.
  + 

**SMOTE TECHNIQUE**

* SMOTE stands for Synthetic Minority Over-sampling Technique, is a widely used technique in machine learning for addressing the class imbalance problem in classification tasks.
* Class imbalance occurs when one class has significantly fewer examples than the others which can lead to biased model performance.
* In many real-world classification problems one class is underperforming compared to others. For instance, in medical diagnosis, the number of patients with a rare disease might be much lower than those without it.
* In our case, the number of companies that are bankrupt is significantly lower than the ones that are not.
* SMOTE addresses the class imbalance problem by generating synthetic samples for the minority class.
* The idea is to create new instances that are similar to the existing minority class instances making the dataset more balanced.
* SMOTE works by selecting a random minority class instance and finding its K Nearest neighbors among the minority class instances.
* It then selects one of the K neighbors randomly and computes the difference between the feature vectors of the selected instance and the chosen neighbor.
* A random value between 0 and 1 is chosen and the synthetic instance is created as a s linear combination of the feature vectors of the selected instance and the neighbor.
* The synthetic instance is added to the dataset, effectively increasing the size of the minority class.

**SMOTE APPLICATION**

* I have imported the required libraries for the application of SMOTE.
* ****
* After importing the library, I have defined the SMOTE in a new variable.
* 
* Here, sampling strategy = 1 means that both the records in the classes will have an equal number of records.
* random\_state = 42, is the seed of random generation of values. I have kept random\_state = 42 across programs to have a constant generation of values.
* Before applying SMOTE, the number of records were as follows.
* ****
* After applying SMOTE, number of records are as follows.
* ****
* The records are now 6599 each for both classes and thus our dataset is more balanced.
* Repeating our initial method of KNN on our newly balanced Dataset.
* Applying Train Test Split method
* ****
* After applying the KNN range method to our newly imbalanced dataset, I can see that the graph has changed drastically.
* After taking k = 1, we can apply to our KNN classifier and get accuracy and confusion matrix.
* ****
* I can observe that we have slight dip in accuracy but our confusion matrix is much more balanced after applying SMOTE.
* The evaluation metrics is as follows.
* ****

**RESULTS COMPARED**

|  |  |  |
| --- | --- | --- |
|  | **BEFORE SMOTE** | **AFTER SMOTE** |
| **Number of Records (0/1)** | 6599/220 | 6599/6599 |
| **K value** | 8 | 1 |
| **Confusion matrix** |  |  |
| **Accuracy** | 0.96 | 0.95 |
| **Precision** | Nan – because no TP or FP values | 0.92 |
| **Recall** | 0.00 - because no TP or FP values | 0.99 |
| **F1 Score** | Nan - because no TP or FP values | 0.95 |

**FURTHER WORK**

* I aim to tune more parameters such as test size to check if I am able to improve accuracy even further.
* I also intend to apply different smote techniques such as BorderLINE SMOTE or ADASYN technique to compare the results even further with different sampling ration such as 1,0.5, and 0.25.