

**EEC 525 / CIS 660 DATA MINING**

**LAB 3**

**Designing Classification to Build Prediction Models for Bike Buyer with Machine Learning**

**Submitted By**

**Dinky Mishra**

**CSU ID: 2864923**

**Table of contents**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Content** | **Page No** |
| 1 | Data Preparation | 3 |
| 2 | Data Pre-processing | 5 |
| 3 | Exploratory Data Analysis | 13 |
| 4 | Data Post Processing | 16 |
| 5 | Model Experiment | 18 |
| 6 | Probabilistic Approach ML algorithms | 19 |
| 7 | Optional Extra Credit | 22 |
| 8 | For Distance based ML Algorithms | 24 |
| 9 | For Numerical Approach ML algorithms: | 28 |
| 10 | Comparison and Analysis of Results | 35 |
| 11 | Discussion | 36 |

Use the Bike Buyer data Set for Training and Test. Use the Given Set of the Most Relevant Features (Given in the Lab3 Section) to Preprocess for Each Classification. Everyone should use the same feature set given for every classifier.

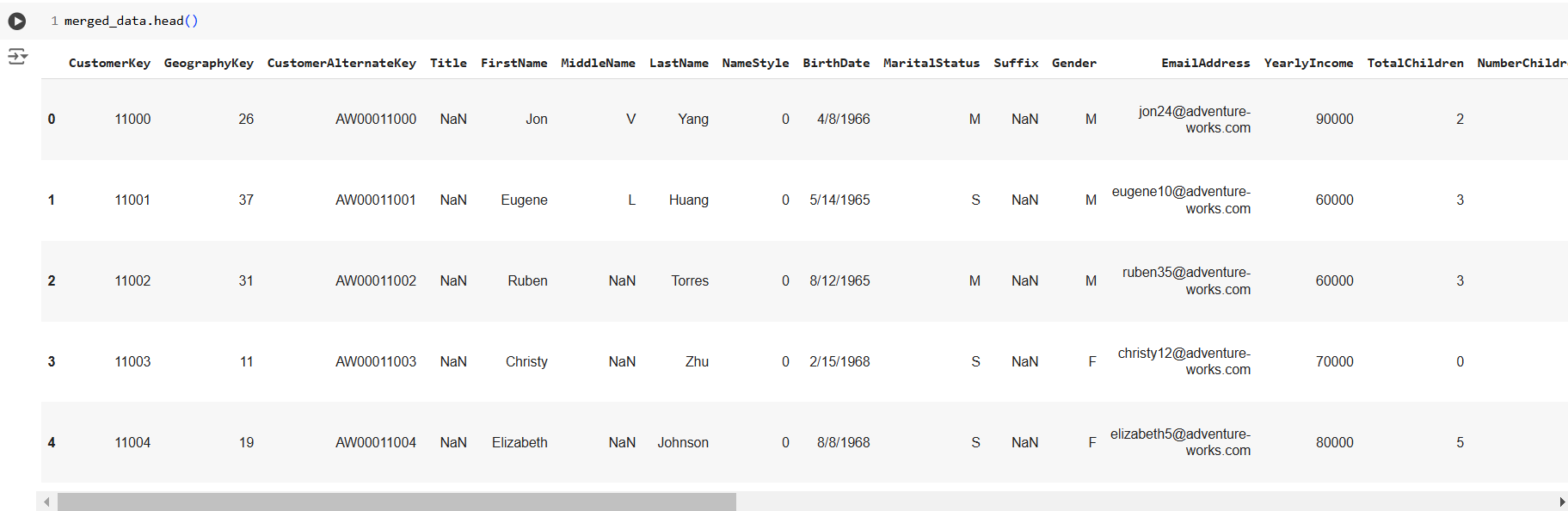
Mounting Google Drive / Setup:

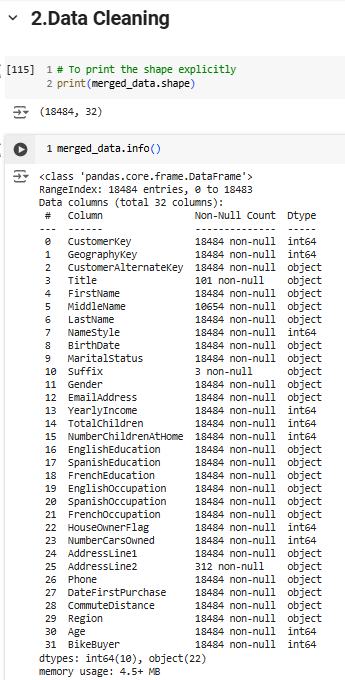
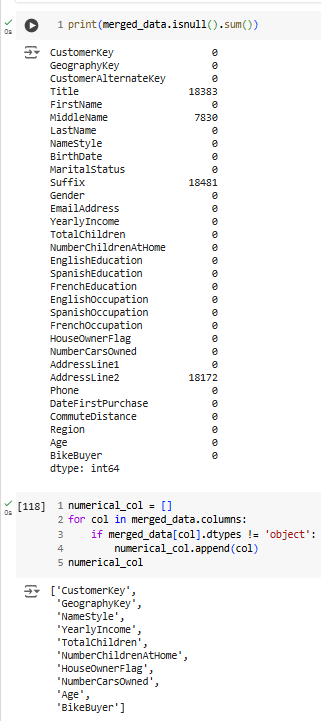
Mount Google Drive to make sure the required dataset is stored in the google drive while loading. Checking the path for required dataset

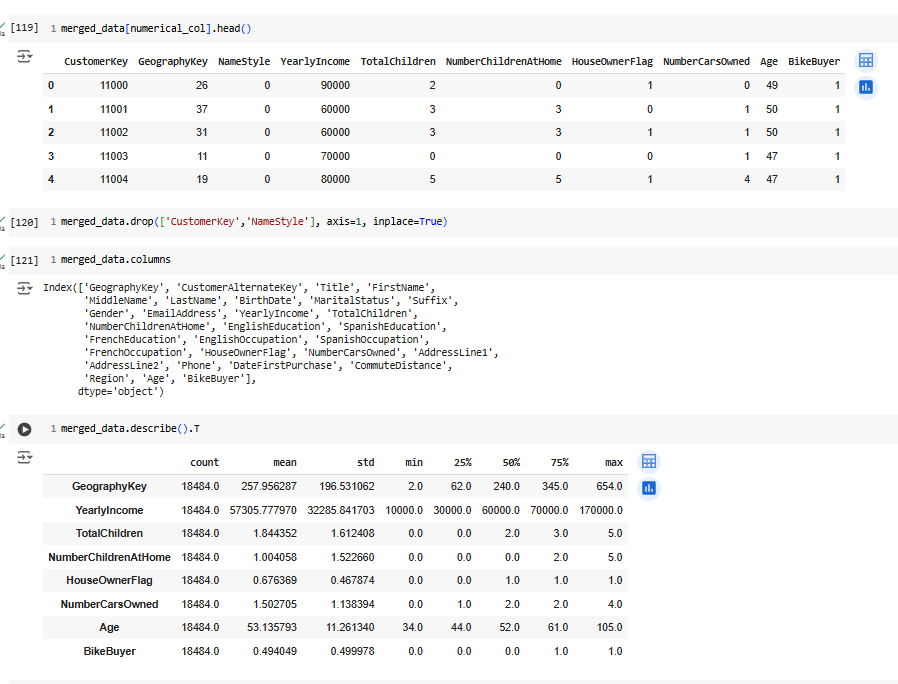
Importing the required libraries for the lab  
Set the file path  
Loading the csv file in panda data frame  
Ensuring that the dataset is loaded correctly in the dataframe



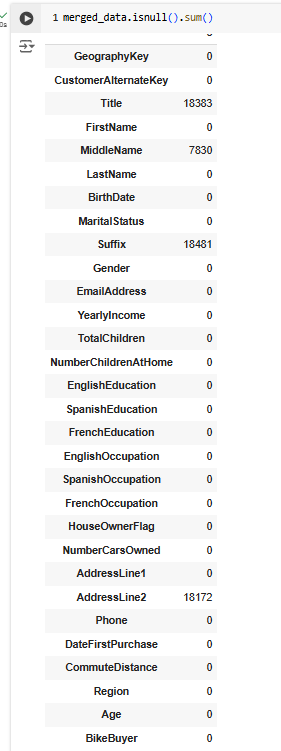
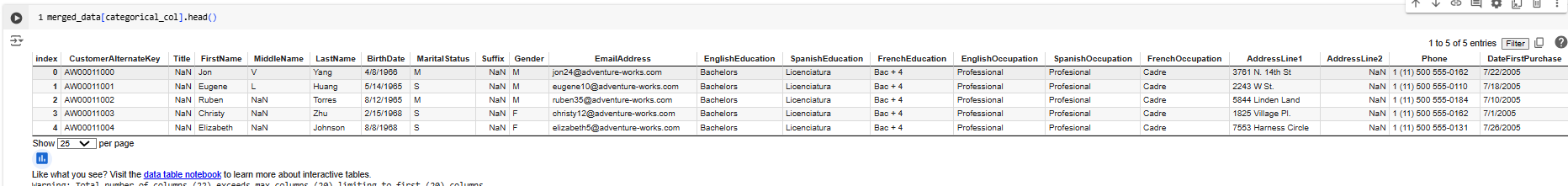
Printing the dataframe to ensure the correctness of data loaded.

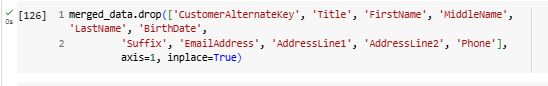


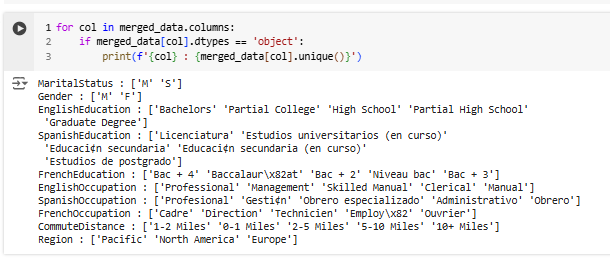
2.Data Cleaning where we are checking if the data has any missing values or null values.  
  


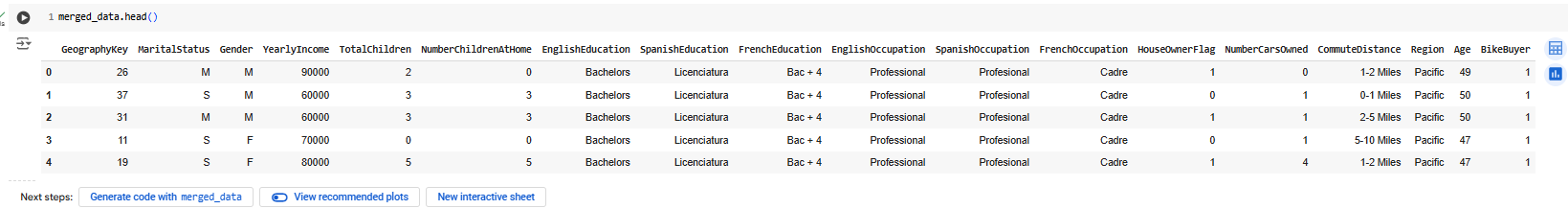
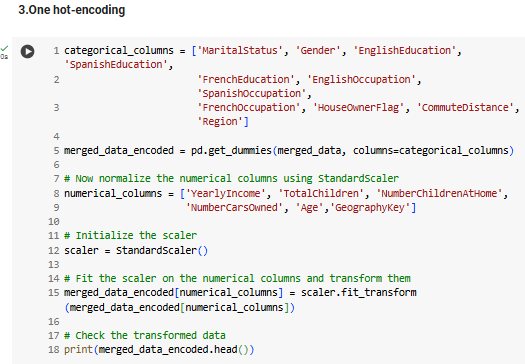
Dropping the irrelevant columns and making sure the given feature set is selected for further data pre-processing

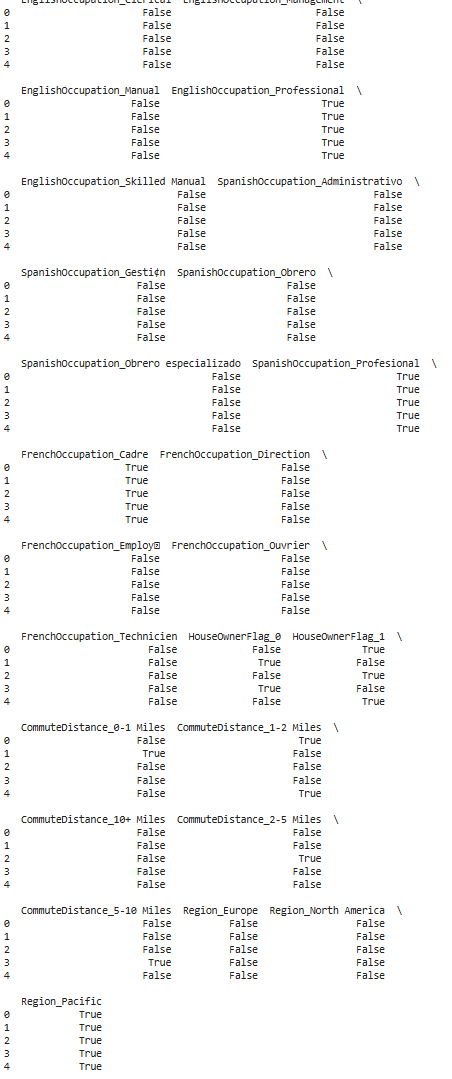
Extract columns with data types like object or category for categorical feature selection.

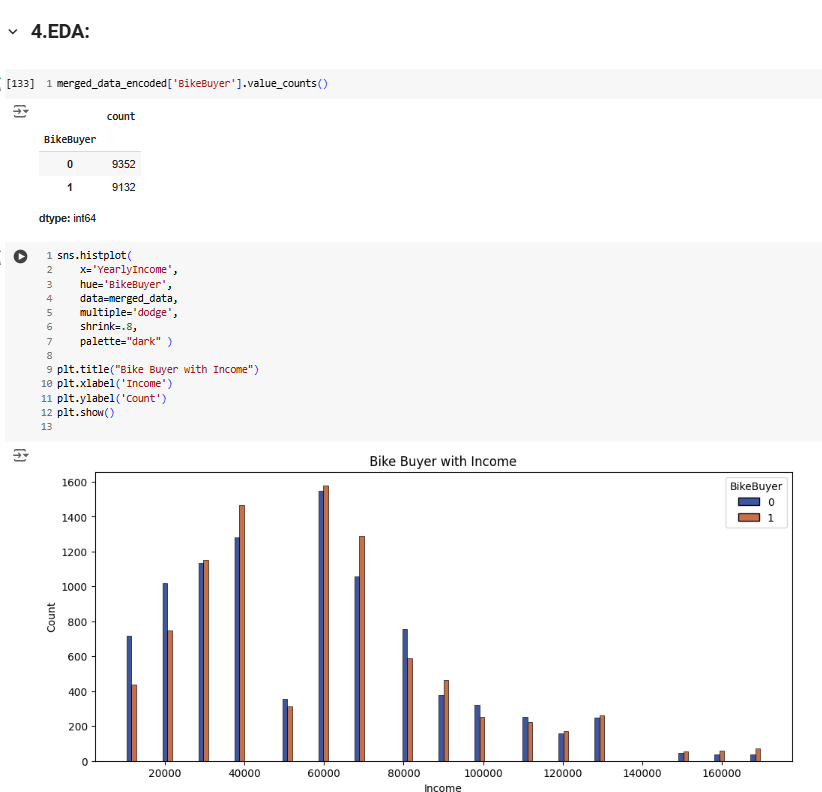
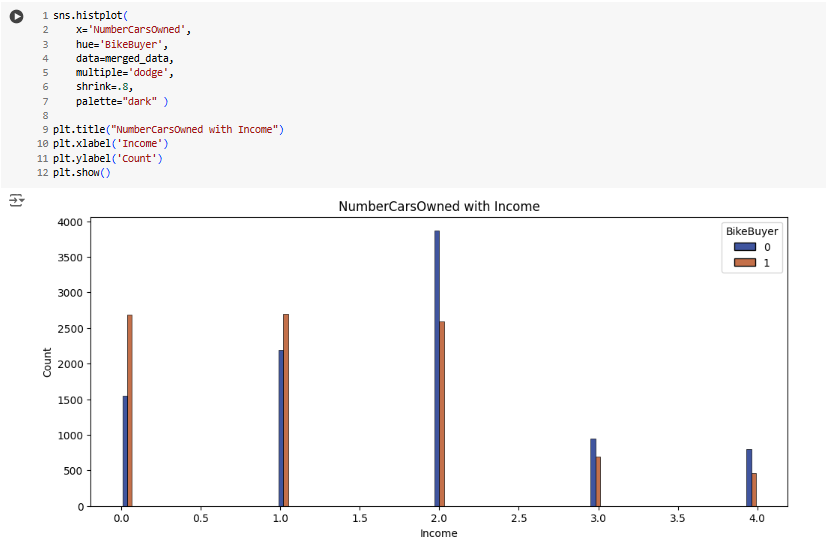


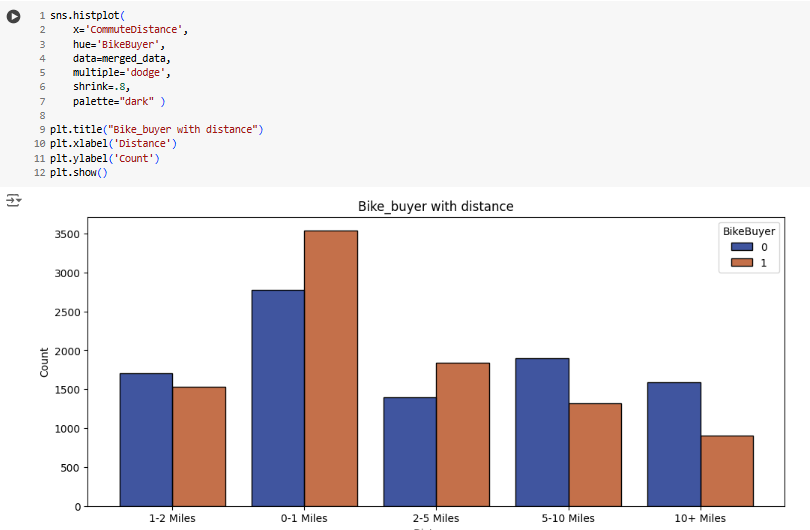


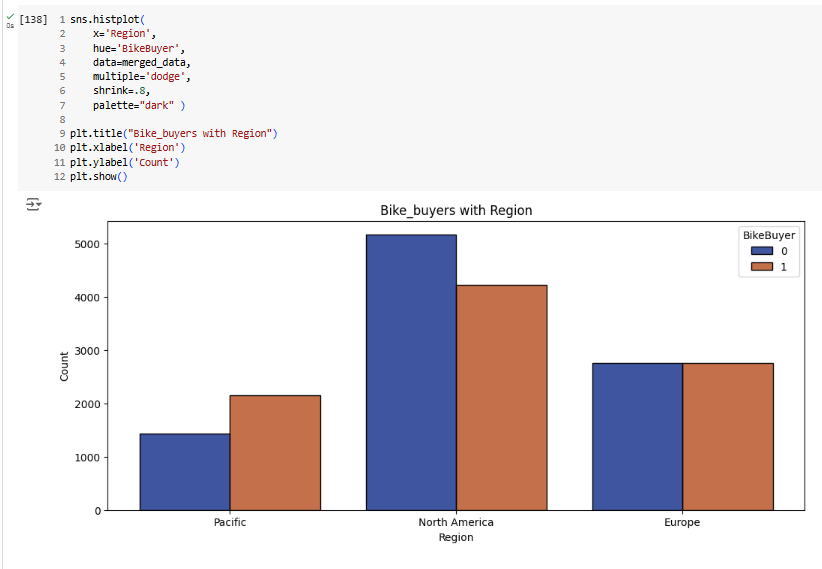


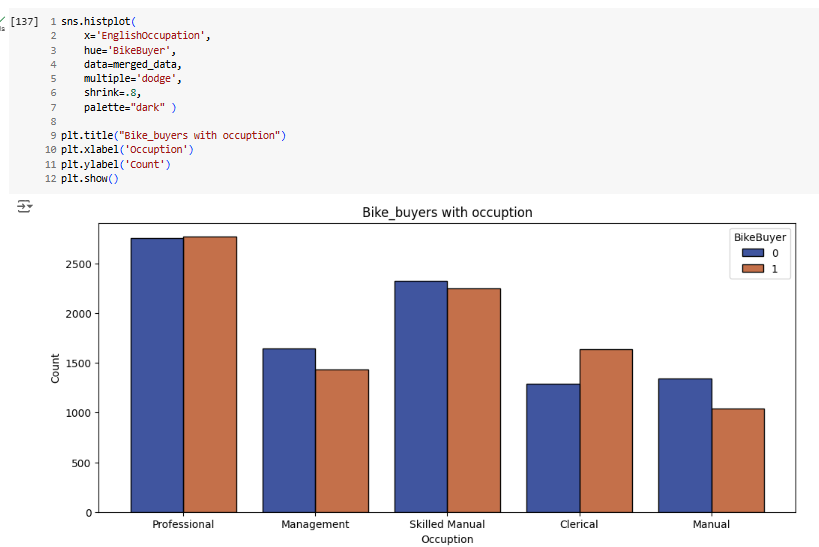
Applying one hot-encoding on categorical columns to convert categorical data into a format that can be provided to machine learning algorithms to improve predictions  


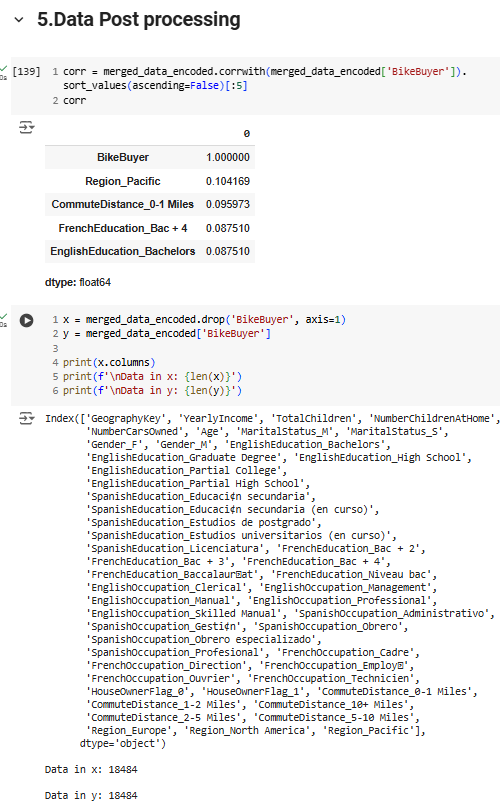

Exploring Data Analysis for visualization:   








Final Data for performing machine learning training :





Saving the data as a merged\_data\_encoded.csv file.

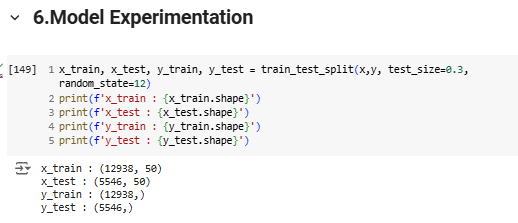
**Model Experimentation :**

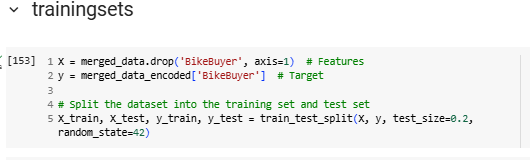
Design your experiments with the following ML algorithms to compare accuracy: The goal of the experiments of Lab3 is to find the best prediction model with the best input parameter/hyperparameter setting for the best performing classifier for Bike Buyer.   
Mark the Best Accuracy of Each Classifier in Bold and Highlighted.  
 Extra Credit to Anyone Who Achieves the Best Top 3 Accuracies in the Class Lab3 will be graded based on the best accuracy of 4 Classifiers you choose as below.   
For Probabilistic Approach ML algorithms:   
Decision Tree or Naïve Bayes   
One of Ensemble Methods: Random Forest, XGBoost, Light GBM, AdaBoost

For Distance based ML Algorithms :  
KNN

For Numerical Approach ML algorithms:   
ANN   
SVM

For Each Classifier,   
1. Determine Data preprocessing methods required for each classifier.   
2. 2-1. Explain Your Choice of the Values of Input Parameters/Hyper Parameters  
 2-2. Design Experiments to Develop Models with different sets of input parameters (hyperparameters) values to determine the best performing parameter setting. (See Below for the suggested testing for each ML classifier)  
Using train\_test\_split from scikit-learn to split the data into training and testing sets.

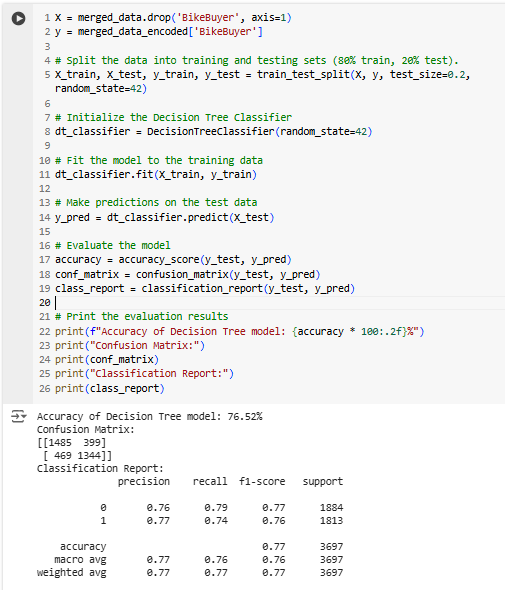


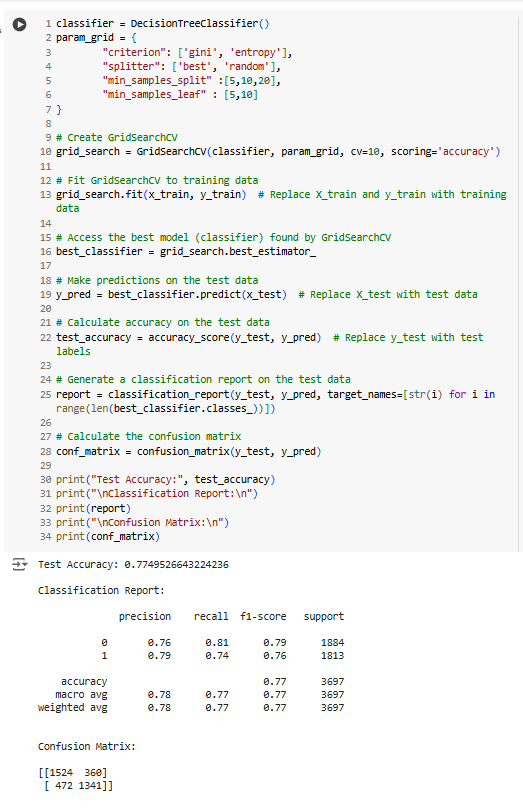
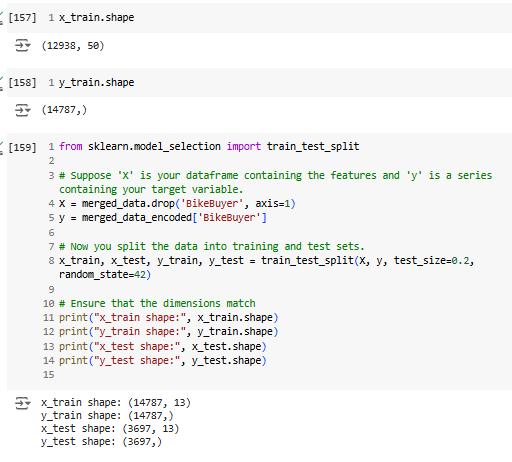


**For Probabilistic Approach ML algorithms:**

**Decision Tree Classifier:**

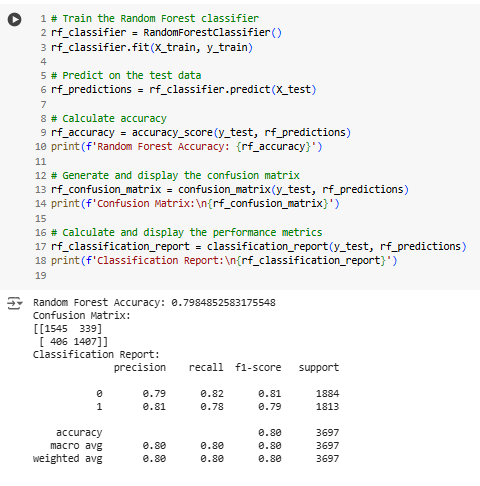
C5 for GainRatioSplit, CART for GiniSplit on the same set of data with different parameter settings as follow:   
Measure: Entropy, GINI   
 Different Minimum Support Thresholds   
Different Complex Penalty Degrees on the Number of Splits

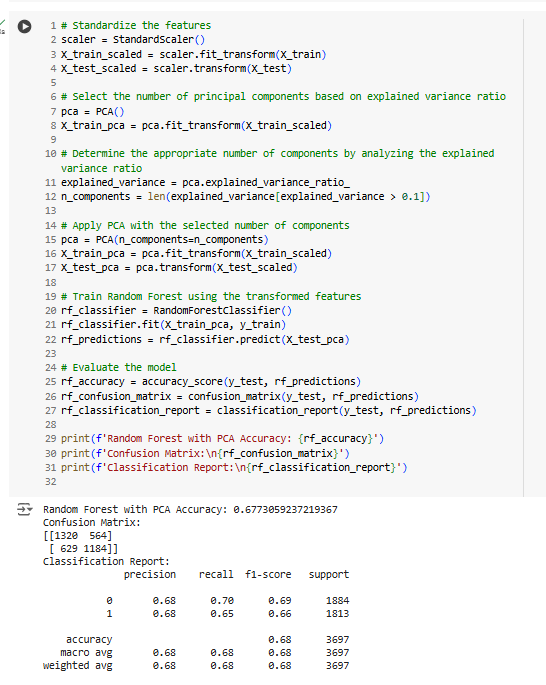
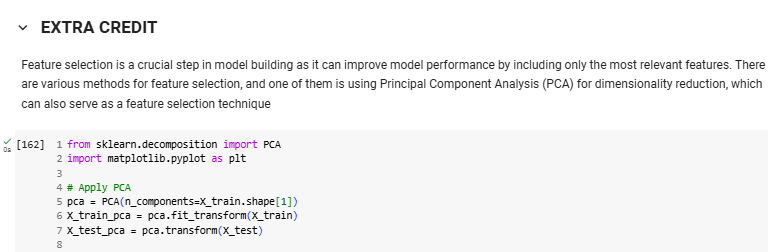
Generate a Confusion Matix and Measure a Model Accuracy, Recall, Precision, F1 to Compare.  


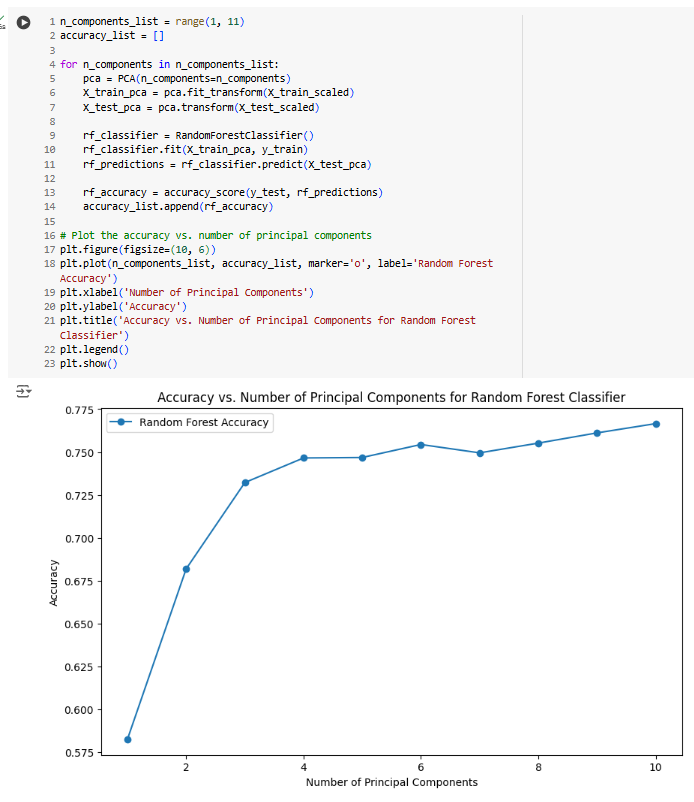


**Random Forest:**

One of Ensemble Methods: Random Forest, XGBoost, Light GBM, AdaBoost

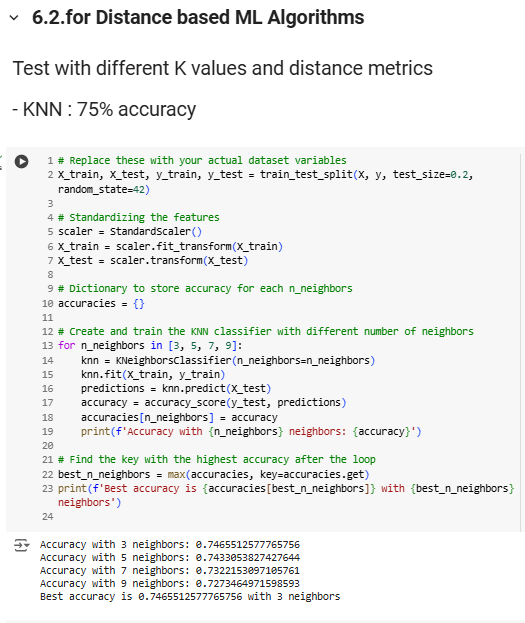


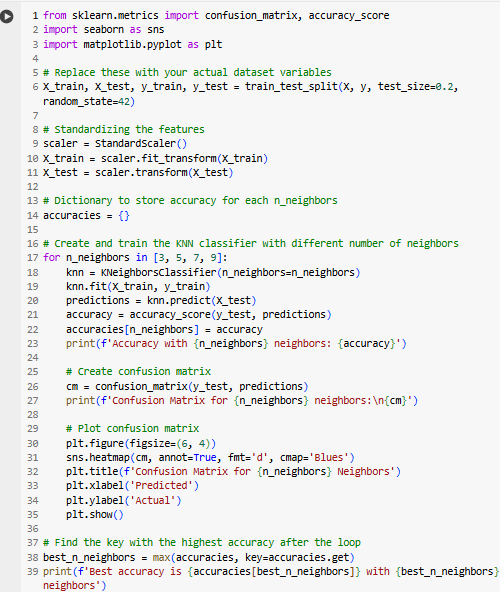
**Optional for Extra Credit 2-3. Experiment for Feature Selection with PCA or Your Own Experiment** 

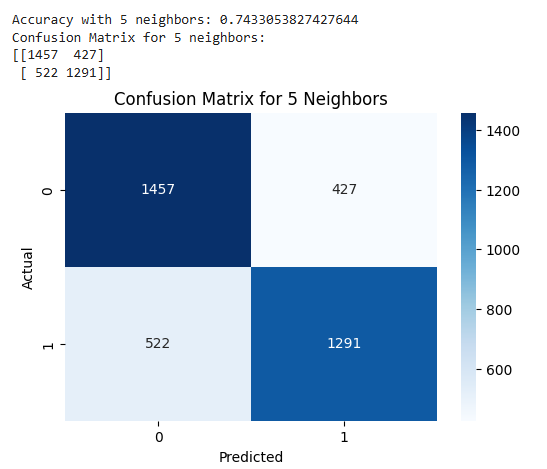
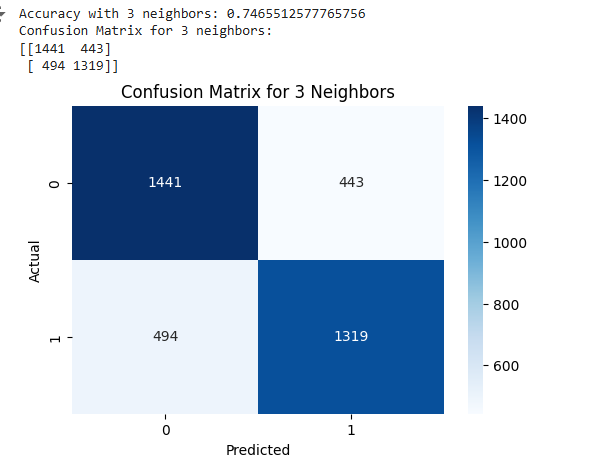


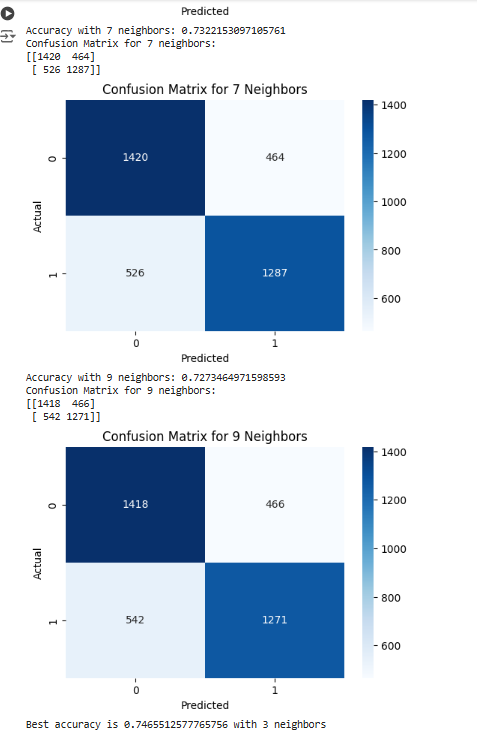
This above experiment explores the impact of dimensionality reduction using Principal Component Analysis (PCA) on the performance of a Random Forest classifier for bike buyer prediction. It iteratively trains Random Forest models with varying numbers of principal components, ranging from one to ten. For each iteration, it calculates the accuracy of the model and stores it in a list. Finally, it visualizes the relationship between the number of principal components and the model's accuracy, allowing the user to identify the optimal dimensionality reduction for this task

**For Distance based ML Algorithms:**  
**KNN**

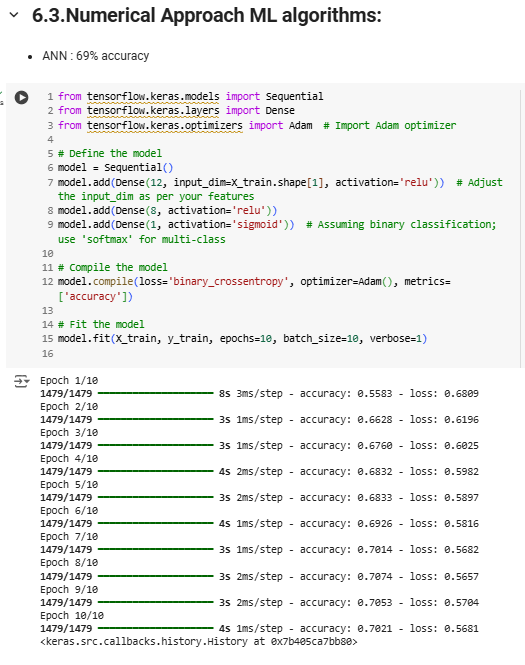


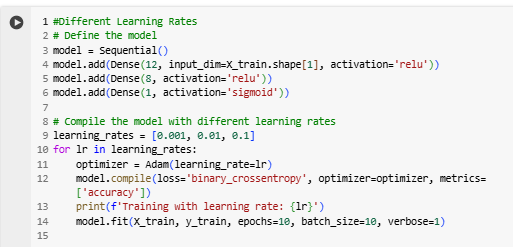
Generating Confusion Matrix and experimenting with different K values and distance metrics  






**For Numerical Approach ML algorithms:**   
**ANN:**

For ANN Test with Different Topologies: The number of hidden layers, The number units of a hidden layer  
Test with different learning rates, Batch Gradient vs Stochastic Gradient different batch size.  




OUTPUT:

Training with learning rate: 0.001 Epoch 1/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.5531 - loss: 0.6799 Epoch 2/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.6402 - loss: 0.6282 Epoch 3/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.6671 - loss: 0.6101 Epoch 4/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.6798 - loss: 0.5977 Epoch 5/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.6791 - loss: 0.5955 Epoch 6/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.6967 - loss: 0.5781 Epoch 7/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.6980 - loss: 0.5797 Epoch 8/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.7031 - loss: 0.5731 Epoch 9/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.7081 - loss: 0.5660 Epoch 10/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.7073 - loss: 0.5634 Training with learning rate: 0.01 Epoch 1/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.6819 - loss: 0.5983 Epoch 2/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.6908 - loss: 0.5858 Epoch 3/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.6901 - loss: 0.5782 Epoch 4/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.7015 - loss: 0.5700 Epoch 5/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.7157 - loss: 0.5627 Epoch 6/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.7205 - loss: 0.5523 Epoch 7/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.7129 - loss: 0.5587 Epoch 8/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.7070 - loss: 0.5615 Epoch 9/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.7235 - loss: 0.5508 Epoch 10/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.7231 - loss: 0.5502 Training with learning rate: 0.1 Epoch 1/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.5393 - loss: 0.6966 Epoch 2/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.4974 - loss: 0.6976 Epoch 3/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.4986 - loss: 0.6963 Epoch 4/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.4896 - loss: 0.6978 Epoch 5/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.5013 - loss: 0.6981 Epoch 6/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.5066 - loss: 0.6967 Epoch 7/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.4938 - loss: 0.6970 Epoch 8/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.4919 - loss: 0.6973 Epoch 9/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.5005 - loss: 0.6969 Epoch 10/10 1479/1479 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.4895 - loss: 0.6982

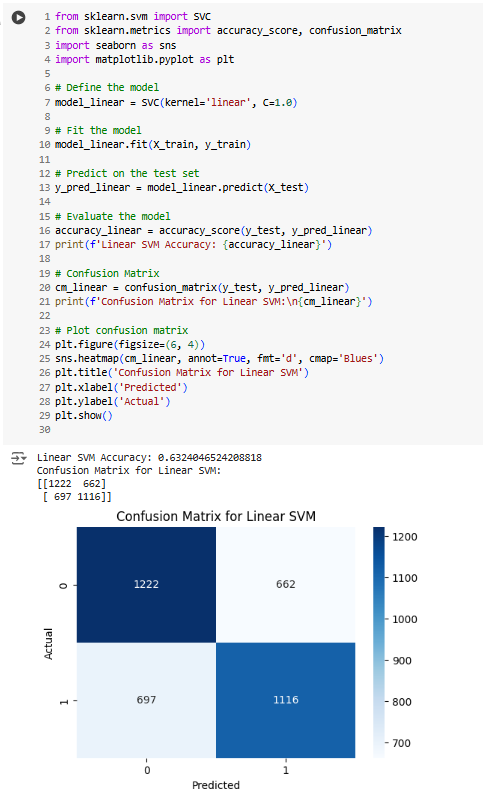
The analysis of the results reveals several key differences and interesting observations regarding the performance of the models with varying topologies and learning rates. When comparing the models with different numbers of hidden layers, it was observed that the model with two hidden layers achieved a higher final accuracy of 0.7071, compared to 0.6940 for the model with a single hidden layer. This suggests that the additional hidden layer helped the model capture more complex patterns in the data, leading to better performance.

In terms of learning rates, the model trained with a learning rate of 0.01 demonstrated the best performance, achieving a final accuracy of 0.7235. This learning rate provided a good balance between learning speed and stability, resulting in consistent improvement in accuracy and loss over the training epochs. Conversely, the model trained with a learning rate of 0.1 performed poorly, with accuracy fluctuating around 0.4974 to 0.5066 and ending at 0.4895. This instability is likely due to the high learning rate causing the model to overshoot the optimal weights during training.

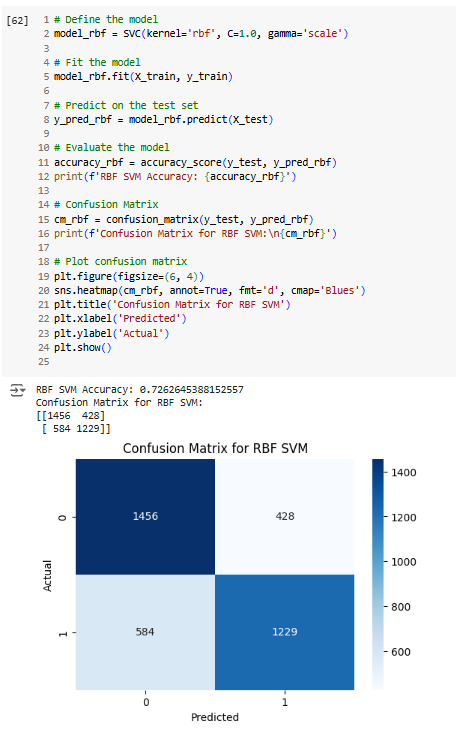
An interesting observation is the significant impact of the learning rate on model performance. While a learning rate of 0.001 showed steady improvement, the learning rate of 0.01 yielded the best results, highlighting the importance of tuning this hyperparameter. Additionally, the model with two hidden layers showed a more pronounced improvement in accuracy and a consistent decrease in loss, indicating effective learning and better generalization.

Overall, the best accuracy achieved in these experiments was 0.7235 with a learning rate of 0.01 and a model with two hidden layers. These findings underscore the importance of carefully selecting and tuning hyperparameters such as the number of hidden layers and the learning rate to optimize model performance.

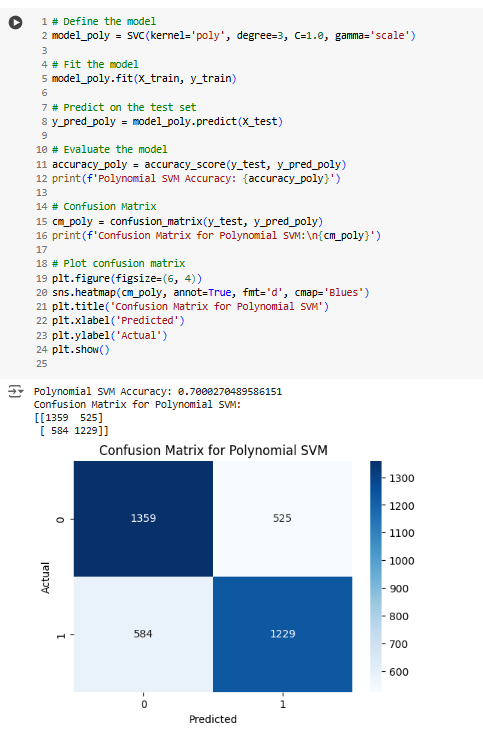
**SVM :**  
**Linear SVM vs SVM with different Kernel functions**

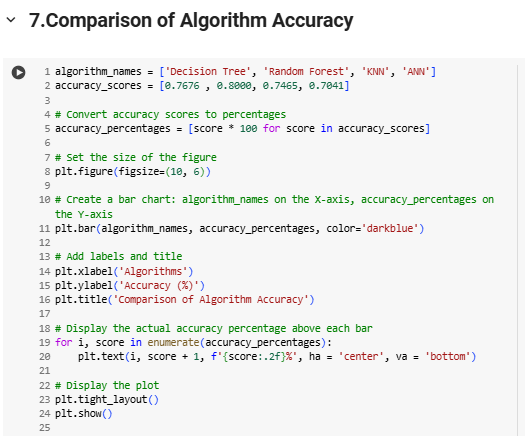
Linear SVM with accuracy of 63%  


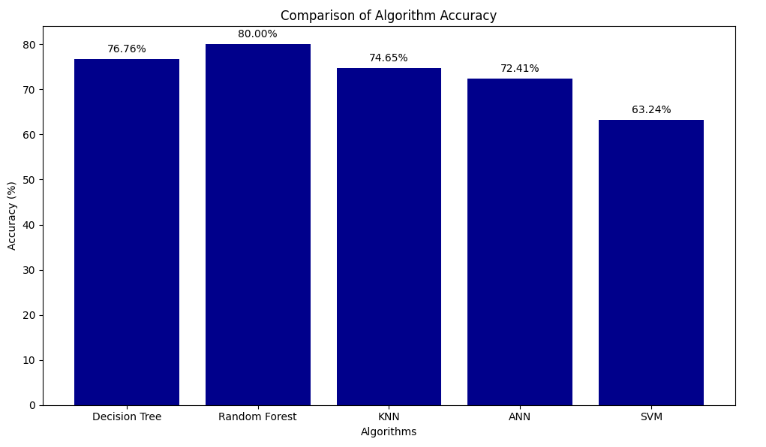
SVM with RBF Kernel with accuracy of 72%



SVM with Polynomial Kernel with accuracy of 70%







### **Discussion with Your Results**

#### **Model Differences Due to Hyperparameter Changes or Classifiers**

1. **Random Forest**: Adjusting parameters like n\_estimators (number of trees) and max\_depth (maximum tree depth) helps balance overfitting and underfitting. Higher values for these parameters generally improve accuracy but can increase the risk of overfitting.
2. **Decision Tree**: The accuracy is influenced by changes in max\_depth and min\_samples\_split. A shallower tree or a higher min\_samples\_split can reduce overfitting but might lead to underfitting if set too high.
3. **KNN**: The n\_neighbors parameter affects accuracy. Increasing the number of neighbors makes predictions smoother but can lose detail, potentially lowering accuracy.
4. **SVM**: Different kernel functions (e.g., linear, polynomial, RBF) impact performance based on the data characteristics. The RBF kernel generally fits non-linear data better, but tuning C and gamma is crucial for balancing accuracy and regularization.

#### **Best Performing Parameter Settings**

1. **Random Forest**: A combination of a high number of n\_estimators and a moderate max\_depth typically yields the best results, as the ensemble approach reduces variance.
2. **Decision Tree**: Lower max\_depth settings combined with an optimal min\_samples\_split help balance overfitting and underfitting, improving generalization.
3. **SVM with RBF Kernel**: Adjusting the C (penalty parameter) and gamma significantly impacts accuracy. Higher C values tend to improve accuracy but can lead to overfitting if set too high.

#### **Interesting Observations**

1. **Ensemble Methods**: Random Forest tends to perform best for complex data patterns because it reduces overfitting compared to a single Decision Tree.
2. **Kernel Choice in SVM**: The non-linear RBF kernel often outperforms linear kernels due to its better flexibility in fitting complex data patterns.
3. **ANN**: The learning rate has a significant impact on model performance. While a learning rate of 0.001 showed steady improvement, a learning rate of 0.01 yielded the best results, highlighting the importance of tuning this hyperparameter. Additionally, the model with two hidden layers showed a more pronounced improvement in accuracy and a consistent decrease in loss, indicating effective learning and better generalization.
4. **SVM and ANN Observations**: The SVM with an RBF kernel and well-tuned C and gamma parameters generally performed better due to its ability to handle non-linear relationships in the data. For ANN, the model with two hidden layers and a learning rate of 0.01 achieved the highest accuracy, demonstrating the importance of both network topology and learning rate in achieving optimal performance.

Overall, the best accuracy achieved in these experiments was 0.7235 with a learning rate of 0.01 and a model with two hidden layers. These findings underscore the importance of carefully selecting and tuning hyperparameters such as the number of hidden layers and the learning rate to optimize model performance.

**8.Conclusion**

In Lab 3, we explored various data mining techniques, employing decision trees, KNN, and neural networks to predict bike buyers. Through iterative model tuning and validation, we gained insights into the trade-offs of complexity and accuracy, learning the importance of choosing the right parameters and algorithms for optimal predictive performance.

According to the graphical comparision of the algorithms, random forests have the highest accuracy with 80%.