Statistical Data Mining - 1

EAS 506

Project Report Group – 2

Abstract

In this project, we examine potential factors that lead to income bias for the 50K income bracket using various classification methods to evaluate the traditional US Adult Income Dataset (income 50K per year). We train the dataset and produce robust models over cross validation. We also observe estimation stability using bootstrap.

1. Introduction:

The US Adult income dataset for this project helps us to classify whether the income of a person is above 50k per year of below 50k per year. We used US Adult Income Dataset to train various classification modele, namely: Logistic Regression, Gaussian Naive Bayes, K – Nearest Neighbor, Support Vector Classifier, Decision Tree, Random Forest. We then used cross validation for comparing the performance of each of the six models and understood the predictor contributors of each feature for each model. We also used bootstrap to understand the estimation stability of each of the models we used on the data.

2. Data Description:

Our data source is from Kaggle; data set name “Adult Income Data” which has the records of individuals whose income is more than 50k or less than 50k per annum. There is a total of ‘48842 rows and ‘15’ columns in the raw dataset:

- age: age of the individual.
- workclass: employment status of an individual
- fnlwgt: final weight of the record
- people represented by this row.
- education-num: education level, in ascending positive integer
- value.
- education: The level of education.
- marital-status: marital status of a person.

- occupation: category of the occupation.
- relationship: relationship in terms of the family.
- race: individuals' race
- gender: Male or Female.
- capital-gain: dollar gain of capital.
- capital-loss: dollar loss of capital.
- hours-per-week: working hours per week.
- native-country: country at birth.
- income: True if greater than equal to 50K, otherwise False (< 50K per year).

```
[ ] df.shape

(48842, 15)
```

Link: https://www.kaggle.com/datasets/wenruliu/adult-income-dataset

3.Data Cleaning/ Preprocessing:

Converting object type to categorical: we found the features that are objects and converted them to categorical features so as to make visualization of those features easy.

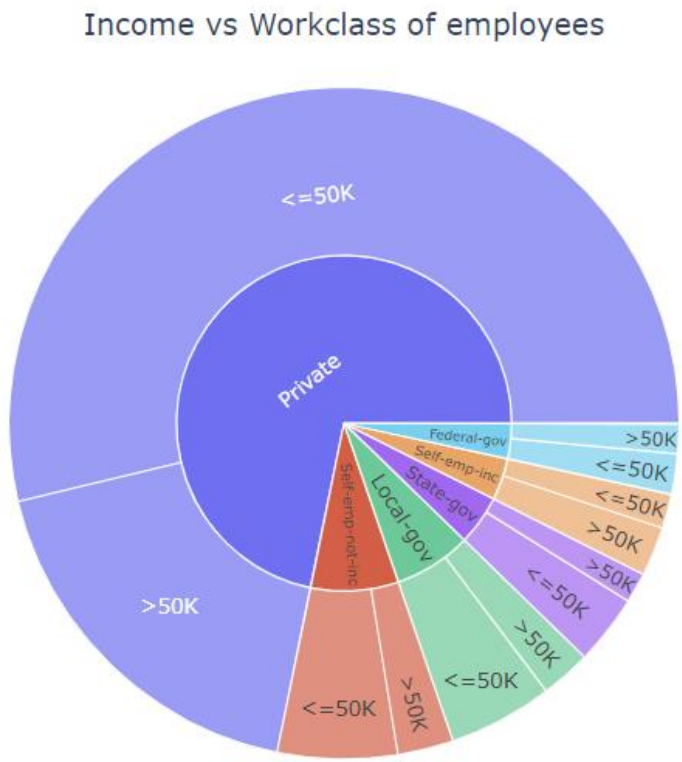
Dropping rows with '?' values: After calculating the sum of '?' values in each column, we converted those rows into null values and dropped them.

After dropping the rows the cleaned dataset have 45222 rows and 15 features.

```
[ ] df.shape

(45222, 15)
```

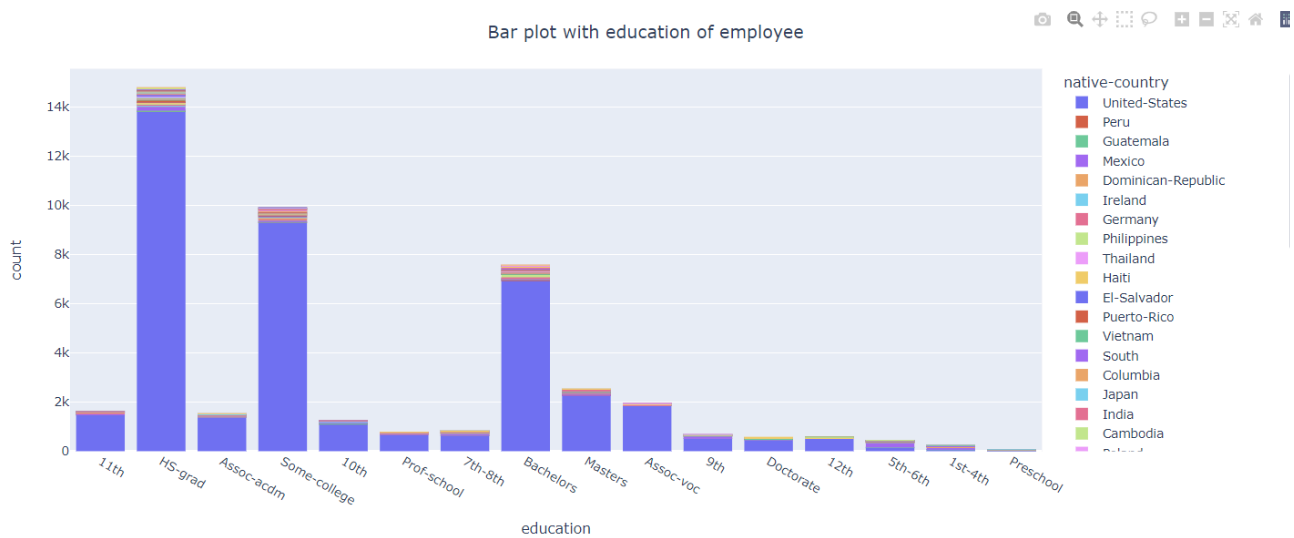
4. Visualization:

Pie Chart of Income vs Workclass:



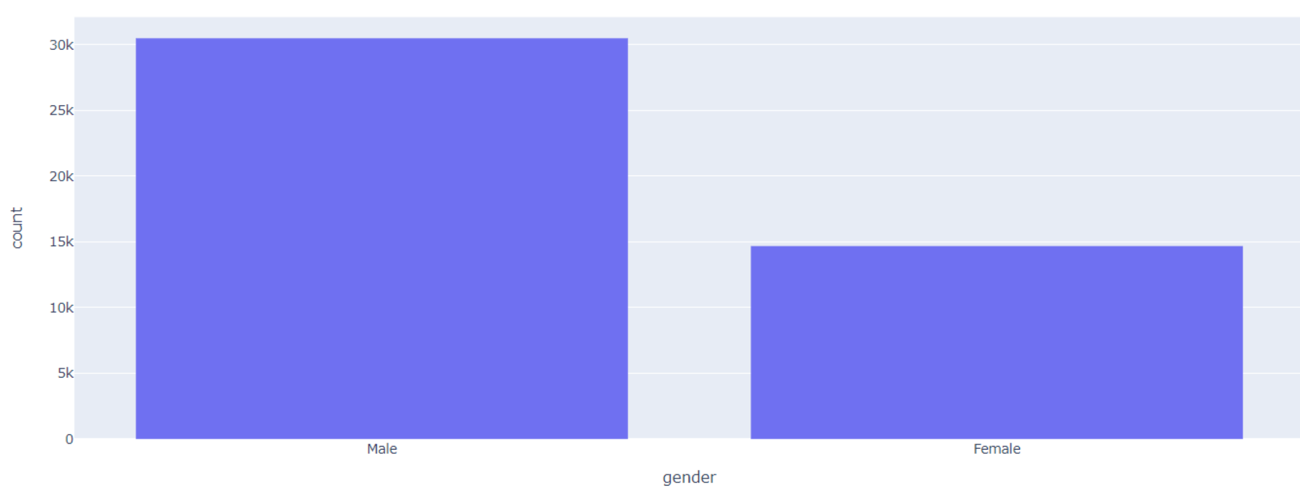
From the chart it is clear that there are a greater number of people working in private sector. Also, there are people working for less than $50k in the private sector. We can also see that there are a greater number of people who are earning more than $50k only in the private sector. There are a smaller number of people working in Federal-gov and their salaries are almost equally split between less than $50k and greater than $50k.

Bar Plot: Education



In the above bar plot the color represents the feature 'native-country'. From the plot it is clear that a greater number of people no matter what their education level is belong to United-States.

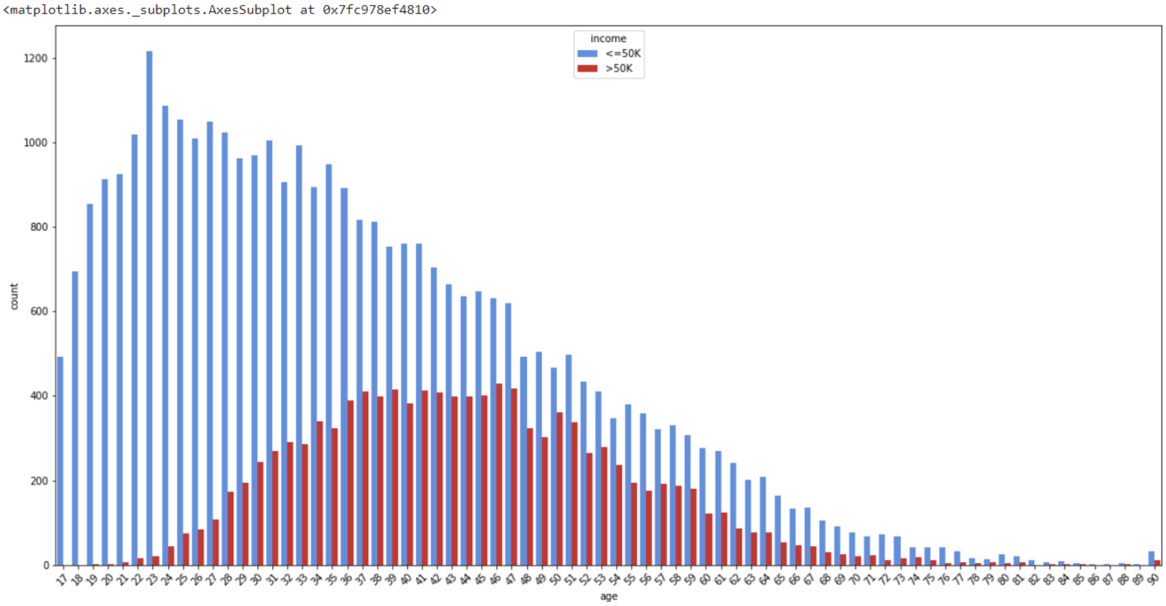
Bar Plot: Gender



From this plot it is clear that there are a greater number of Male employees than Female employees.

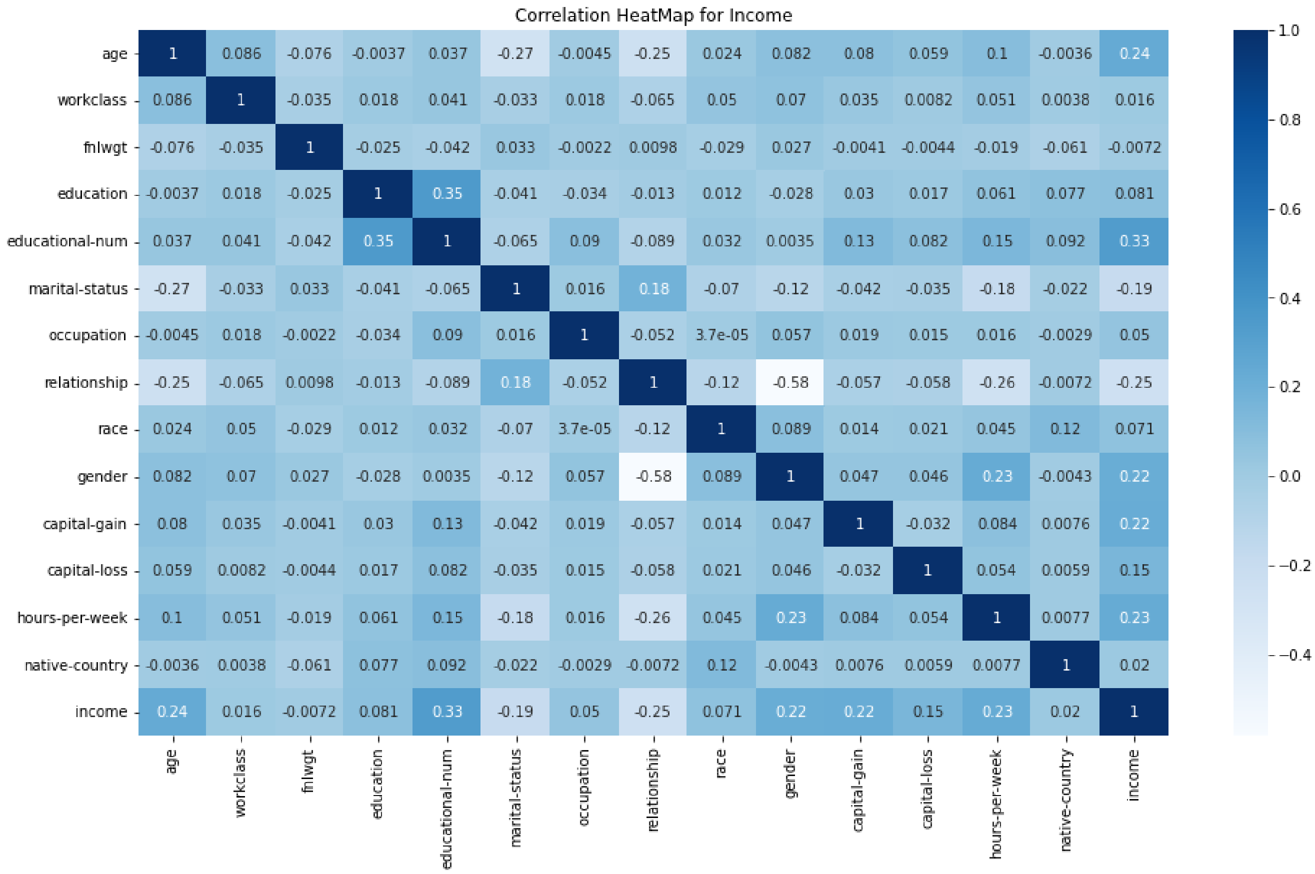
Count Plot: Age vs Income

<matplotlib.axes._subplots.AxesSubplot at 0x7fc978ef4810>



This is very interesting plot. As age grows, there are more people earning more than 50,000$, so we can say that, generally, income is correlated to age.

5. Pairwise association:



- From heat map we understood the correlation between the features of the dataset and the target feature ‘Income’.
- We see that feature like age, educational-num, hours-per-week have positive linear relationship with correlation coefficients of 0.24, 0.33 and 0.23 respectively.
- This means, that an individual’s age is one of the major features because a teenager cannot earn more than $50k, also a greater number of working hours pay more. And education also plays an important role in increasing an individual’s income.

6. Methods:

1. Logistic Regression:

This classification type adopts minimal training data. It predicts the probability of Y being associated with the X input variable.

2. Gaussian Naive Bayes:

It is based on the Bayes theorem, which explains how the likelihood of an event is assessed using information about potential confounding factors in the past.

3. K – Nearest Neighbor:

The K closest neighbors to a given observation site are identified by this classification type. The target variable with the highest ratio is then predicted after evaluating the proportions of each type of target variable using K points.

4. Support Vector Classifier:

Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient. This technique is made feasible by using kernels, which are specialized functions, to expand the space occupied by feature variables. Effective for high dimensional spaces.

5. Decision Tree:

Based on specific feature variables, Decision tree separates a dataset into segments. The mean or mode of the feature variable in question, if it happens to be numerical, serves as the divisions' threshold values.

6. Random Forest:

Random forest processes many decision trees, each one predicting a value for target variable probability. We then arrive at the final output by averaging the probabilities.

7. Results:

1. Logistic Regression:

Training Accuracy: 80.81%

```
[ ] y_predtrain = LRModel.predict(X_train)
    accuracy_score(y_train, y_predtrain)
```

```
0.8081073602656337
```

Testing Accuracy: 81.41%

```
y_pred = LRModel.predict(X_test)
accuracy_score(y_test, y_pred)
```

```
0.8141671278361926
```

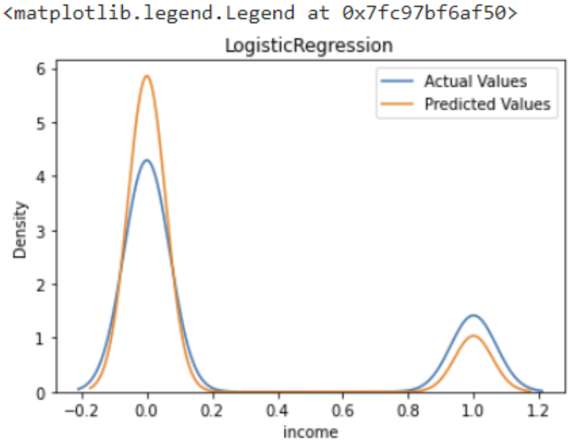
Mean Squared Error:

```
[ ] #mean_squared_error
    from sklearn.metrics import mean_squared_error
    mse = mean_squared_error(y_test, y_pred)
    print(mse)
```

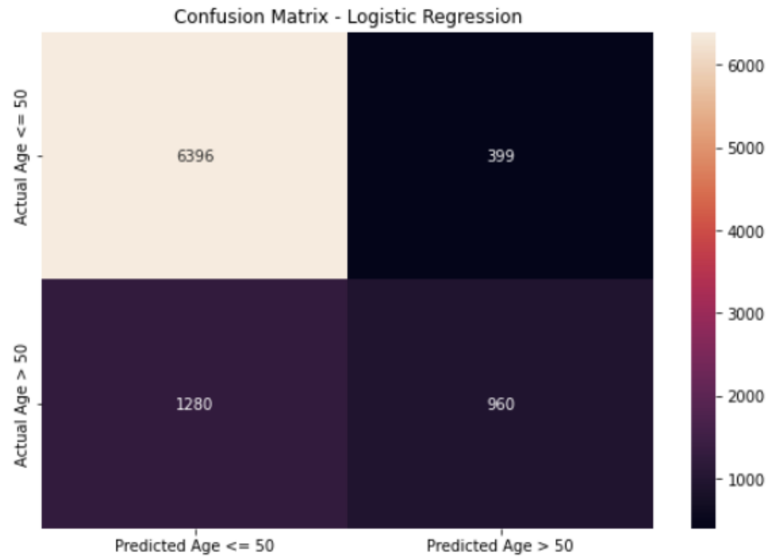
```
0.18583287216380742
```

Distribution Plot: From the plot we can see the comparison between actual value and predicted value.

```
<matplotlib.legend.Legend at 0x7fc97bf6af50>
```



Confusion Matrix:



2. Gaussian Naive Bayes:

Training Accuracy: 79.61%

```
[ ]  GnbModel.score(X_train, y_train)

     0.7961261759822911
```

Testing Accuracy: 80.19%

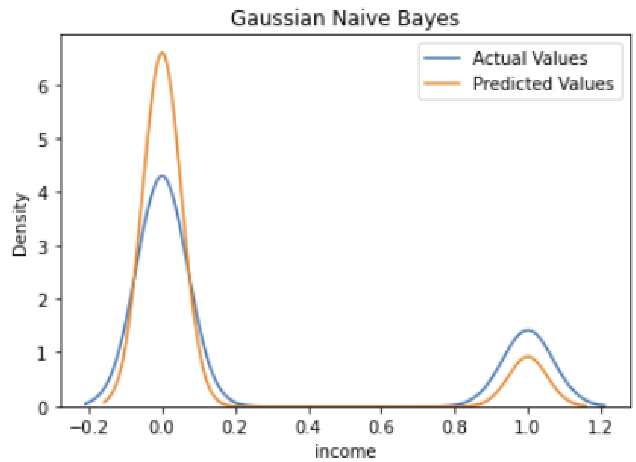
```
GnbModel.score(X_test, y_test)

0.8019922523519646
```

Mean Squared Error:

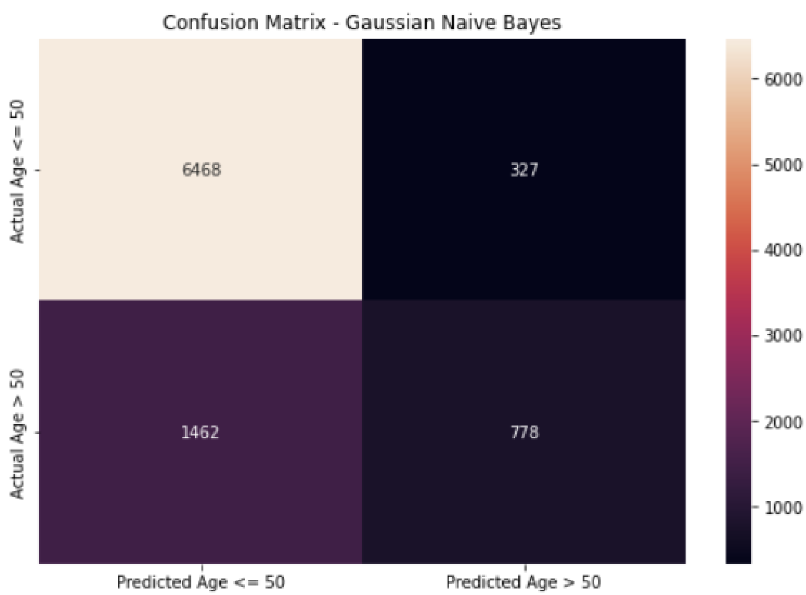
```
[ ]  y_pred = GnbModel.predict(X_test)
     mse = mean_squared_error(y_test, y_pred)
     print(mse)

     0.19800774764803541
```

Distribution Plot: From the plot we can see the comparison between actual value and predicted value.

```
<matplotlib.legend.Legend at 0x7fc9791305d0>
```



Confusion Matrix:



3. K – Nearest Neighbor:

Training Accuracy: 84.15%

```
[ ] KnnModel.score(X_train,y_train)

    0.8415882678472607
```

Testing Accuracy: 82.55%

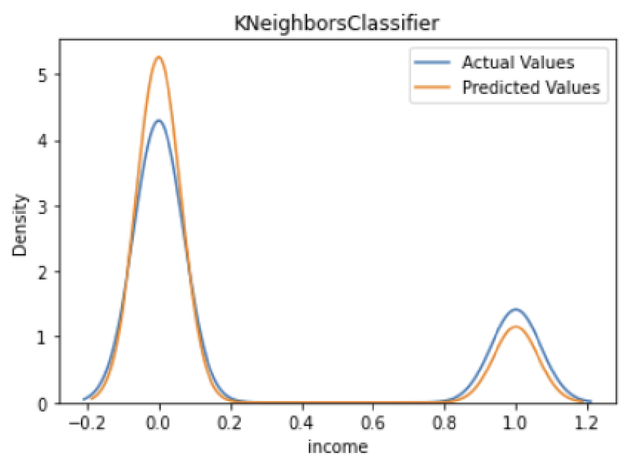
```
[ ] KnnModel.score(X_test,y_test)

    0.8255672385168789
```

Mean Squared Error:

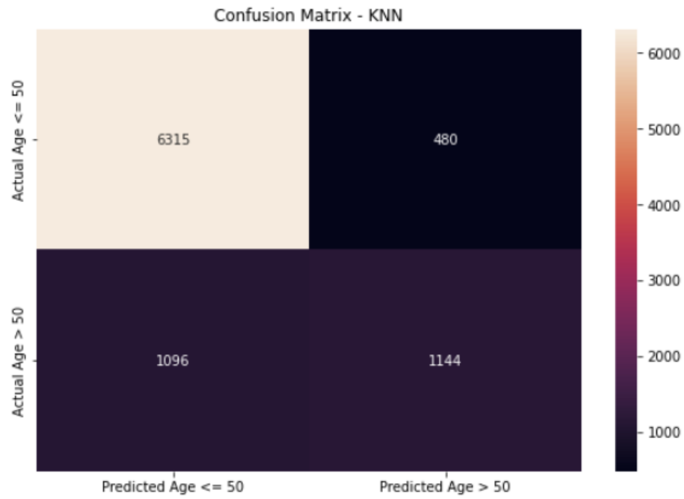
```
[ ] y_pred = KnnModel.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    print(mse)

    0.1744327614831212
```

Distribution Plot: From the plot we can see the comparison between actual value and predicted value.

```
<matplotlib.legend.Legend at 0x7fc978ff5b50>
```



Confusion Matrix:



4. Support Vector Classifier:

Training Accuracy: 80.88%

```
[ ]  y_pred=SVC_Model.predict(X_train)
     accuracy_score(y_train,y_pred)

     0.8088821250691755
```

Testing Accuracy: 81.40%

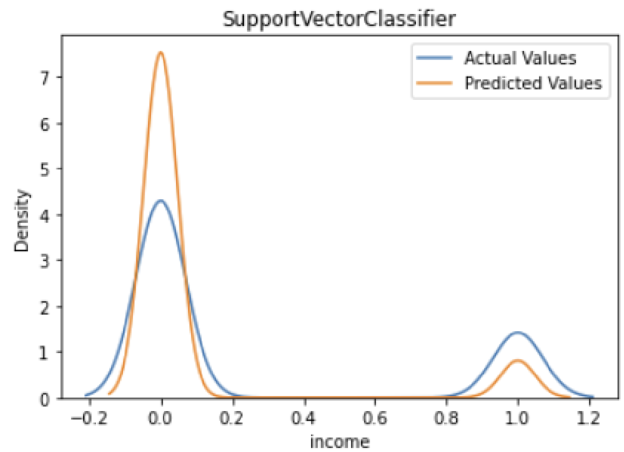
```
[ ]  y_pred=SVC_Model.predict(X_test)
     accuracy_score(y_test,y_pred)

     0.8140564471499724
```

Mean Squared Error:

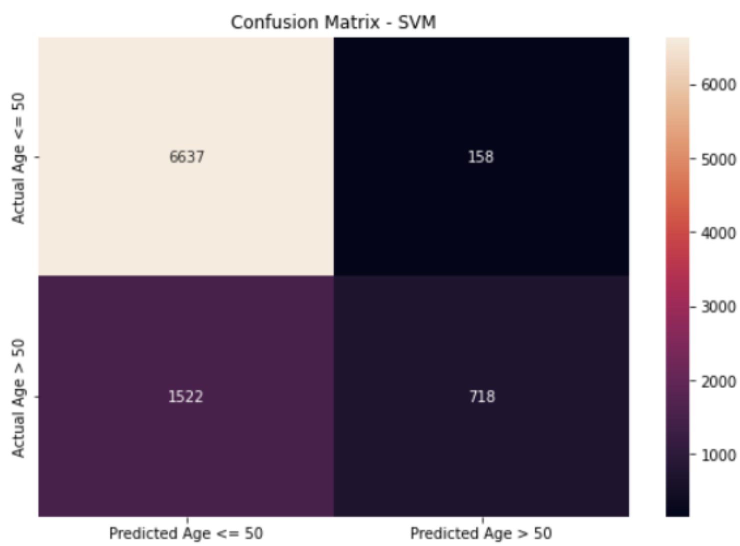
```
[ ]  mse = mean_squared_error(y_test, y_pred)
     print(mse)

     0.18594355285002767
```

Distribution Plot: From the plot we can see the comparison between actual value and predicted value.

```
<matplotlib.legend.Legend at 0x7fc978fc4f50>
```



Confusion Matrix:



5. Decision Tree:

Training Accuracy: 84.4%

```
[ ]  DtreeModel.score(X_train,y_train)

     0.8482014388489209
```

Testing Accuracy: 85.20%

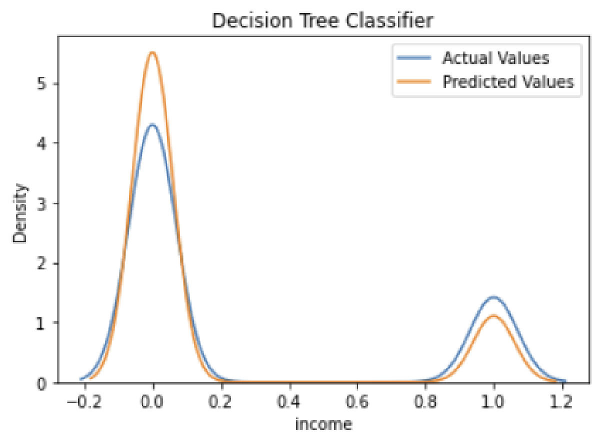
```
[ ] DtreeModel.score(X_test,y_test)

0.8520199225235197
```

Mean Squared Error:

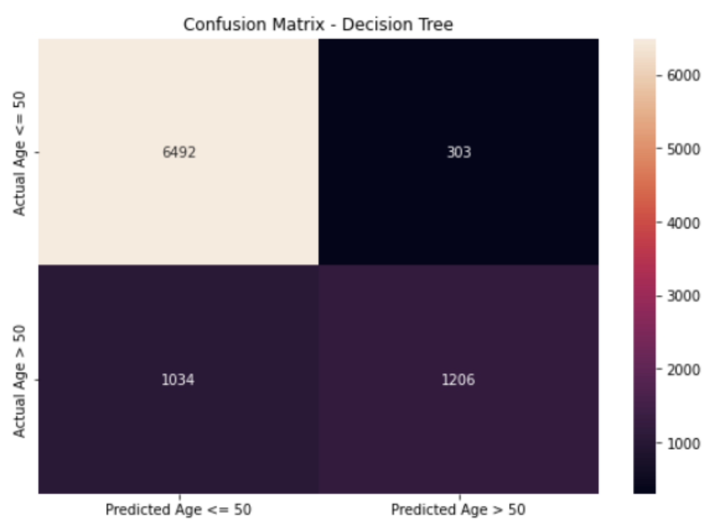
```
[ ] y_pred = DtreeModel.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    print(mse)

0.14798077747648034
```

Distribution Plot: From the plot we can see the comparison between actual value and predicted value.

```
<matplotlib.legend.Legend at 0x7fc9790d4490>
```



Confusion Matrix:



6. Random Forest:

Training Accuracy: 99.9%

```
[ ]  RFModel.score(X_train,y_train)
```

```
0.9999169894853348
```

Testing Accuracy: 85.42%

```
[ ]  RFModel.score(X_test,y_test)
```

```
0.8542335362479248
```

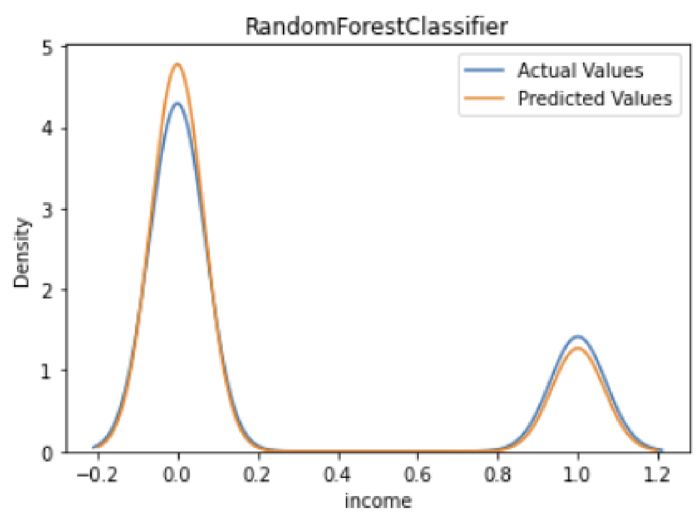
Mean Squared Error:

```
[ ]  y_pred = RFModel.predict(X_test)
     mse = mean_squared_error(y_test, y_pred)
     print(mse)
```

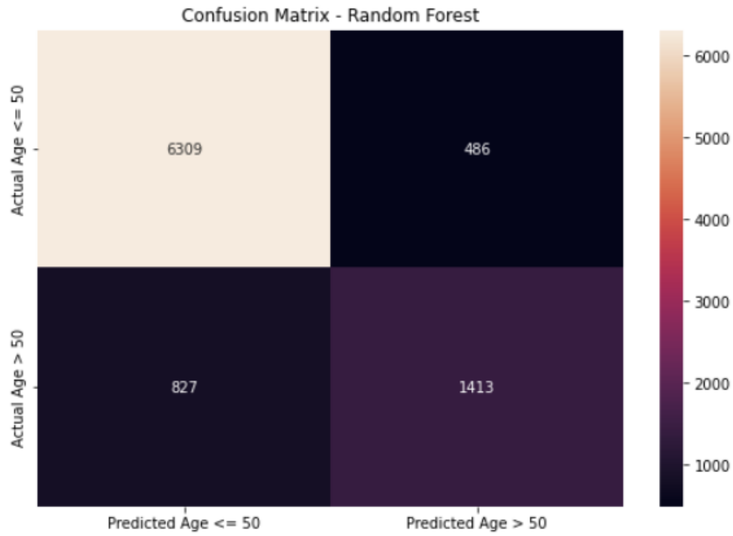
```
0.14576646375207528
```

Distribution Plot: From the plot we can see the comparison between actual value and predicted value.

```
<matplotlib.legend.Legend at 0x7fc97db08e90>
```



Confusion Matrix:



8. Cross Validation:

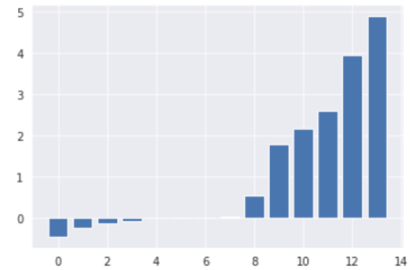
	CV Mean	Std
Logistic Regression	0.809164	0.005204
Gaussian Naive Bayes	0.796923	0.002472
KNN	0.826940	0.003990
Support Vector Classifier	0.810116	0.003223
Decision Tree	0.847549	0.004802
Random Forest	0.854499	0.002919

Cross Validation is performed on all the six models. From the results we can see that Random Forest and Decision Tree have almost similar performance.

9. Predictor Contributors: Feature importance

- Logistic Regression:

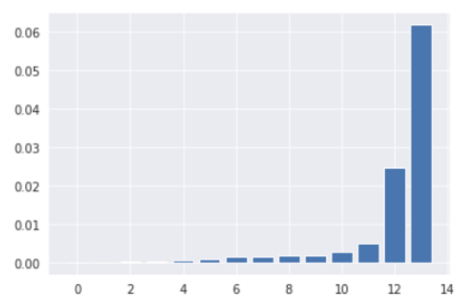
```
Feature : 0 has score  : -0.4601293025577849
Feature : 1 has score  : -0.24182293611044475
Feature : 2 has score  : -0.1305289507922846
Feature : 3 has score  : -0.07238383914170195
Feature : 4 has score  : -0.0019088098359019198
Feature : 5 has score  : 0.0042564869263115625
Feature : 6 has score  : 0.006340738206166303
Feature : 7 has score  : 0.02095855370469415
Feature : 8 has score  : 0.546518551819223
Feature : 9 has score  : 1.7807940313756252
Feature : 10 has score  : 2.164225496183604
Feature : 11 has score  : 2.582710361305024
Feature : 12 has score  : 3.9584947347390003
Feature : 13 has score  : 4.888354594616413
```



From the above graph it is clear that features 10,11,12,13 have high predictor contribution for Logistic Regression.

- Gaussian Naive Bayes

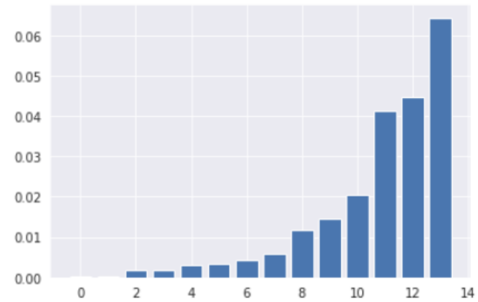
```
Feature: 0  Score: -6.198118428337728e-05
Feature: 1  Score: -4.427227448813298e-05
Feature: 2  Score: 0.00011068068622022142
Feature: 3  Score: 0.0003320420586607309
Feature: 4  Score: 0.00045600442722739667
Feature: 5  Score: 0.0007349197565024613
Feature: 6  Score: 0.0015273934698394553
Feature: 7  Score: 0.0015318206972882776
Feature: 8  Score: 0.001757609297177587
Feature: 9  Score: 0.0018771444382954527
Feature: 10  Score: 0.0027670171555063128
Feature: 11  Score: 0.005011621472053074
Feature: 12  Score: 0.024770337576092927
Feature: 13  Score: 0.06183508577753181
```



From the above graph features 11,12,13 have high predictor contribution for KNN.

- K – Nearest Neighbor

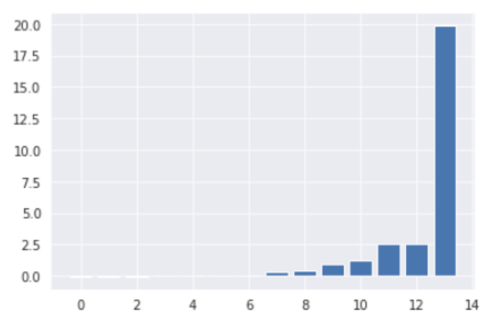
```
Feature: 0  Score: 0.00012396236856666577
Feature: 1  Score: 0.00016380741560595435
Feature: 2  Score: 0.0017974543442168978
Feature: 3  Score: 0.0019037078029883637
Feature: 4  Score: 0.003107913669064754
Feature: 5  Score: 0.003338129496402863
Feature: 6  Score: 0.004316546762589901
Feature: 7  Score: 0.005688987271721069
Feature: 8  Score: 0.011745434421693424
Feature: 9  Score: 0.01460542335362478
Feature: 10  Score: 0.020312119535141095
Feature: 11  Score: 0.04123962368566846
Feature: 12  Score: 0.044595462091864956
Feature: 13  Score: 0.06434975096845597
```



From the above graph feature 13 have high predictor contribution for SVC.

- Support Vector Classifier

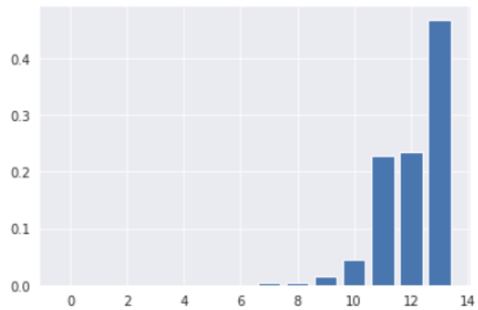
```
Feature : 0 has score  : -0.08260792819874041
Feature : 1 has score  : -0.07834271946774152
Feature : 2 has score  : -0.0598581239013356
Feature : 3 has score  : -0.0035460590461298125
Feature : 4 has score  : 8.692730261827819e-05
Feature : 5 has score  : 0.002035295290625072
Feature : 6 has score  : 0.03347666954630313
Feature : 7 has score  : 0.3527687250101101
Feature : 8 has score  : 0.36953205992796256
Feature : 9 has score  : 0.9428464390985432
Feature : 10 has score  : 1.2127304068820024
Feature : 11 has score  : 2.525832351910006
Feature : 12 has score  : 2.559544238833408
Feature : 13 has score  : 19.85504190851177
```



From the above graph feature 13 have high predictor contribution for SVC.

- Decision Tree

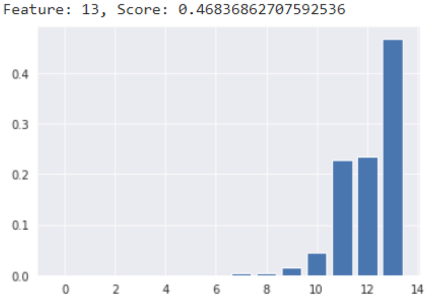
```
Feature: 0, Score: 0.0
Feature: 1, Score: 0.0
Feature: 2, Score: 0.0
Feature: 3, Score: 0.0
Feature: 4, Score: 0.0
Feature: 5, Score: 0.0
Feature: 6, Score: 0.00021270912052782397
Feature: 7, Score: 0.0035920695512579757
Feature: 8, Score: 0.003600674154070468
Feature: 9, Score: 0.015549094579728346
Feature: 10, Score: 0.04464753327446811
Feature: 11, Score: 0.22867376575050297
Feature: 12, Score: 0.23535552649351887
Feature: 13, Score: 0.46836862707592536
```



From the above graph features 13 have high predictor contribution for Decision Tree.

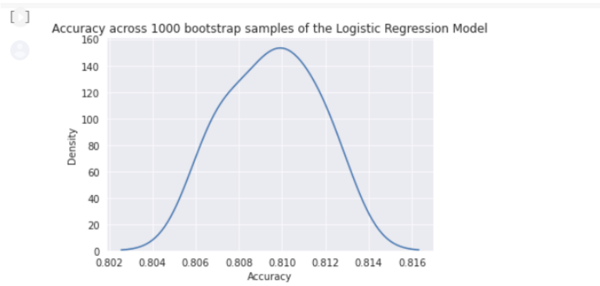
- Random Forest

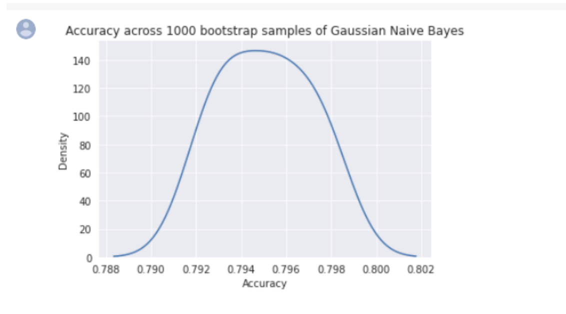
```
Feature: 0, Score: 0.0
Feature: 1, Score: 0.0
Feature: 2, Score: 0.0
Feature: 3, Score: 0.0
Feature: 4, Score: 0.0
Feature: 5, Score: 0.0
Feature: 6, Score: 0.00021270912052782397
Feature: 7, Score: 0.0035920695512579757
Feature: 8, Score: 0.003600674154070468
Feature: 9, Score: 0.015549094579728346
Feature: 10, Score: 0.04464753327446811
Feature: 11, Score: 0.22867376575050297
Feature: 12, Score: 0.23535552649351887
Feature: 13, Score: 0.46836862707592536
```

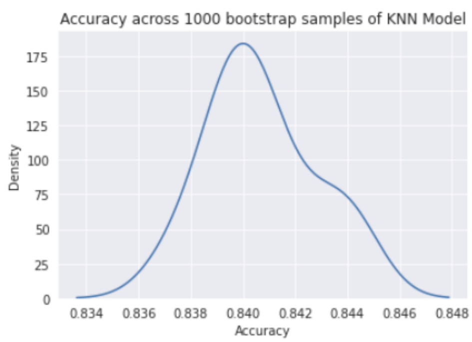


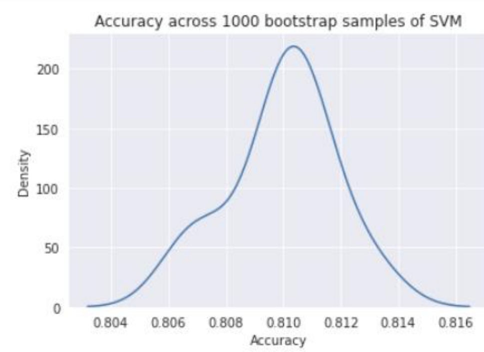
From the above graph features 11,12,13 have high predictor contribution for Random Forest.

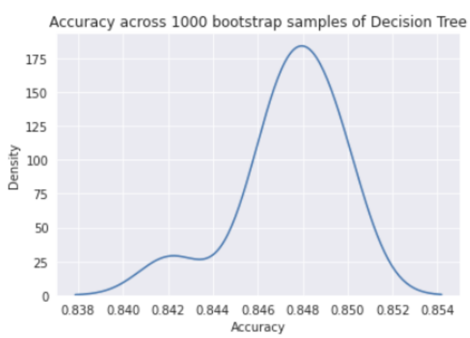
10. Estimation Stability via bootstrap:

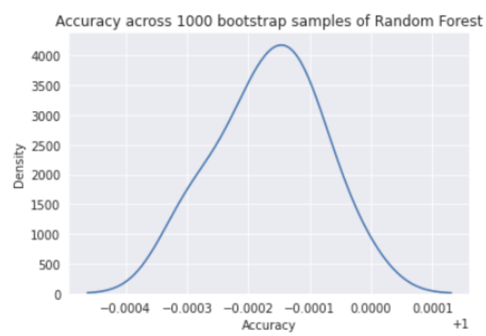












From the above graphs, Random Forest model has the best result of all the 6 models we applied as the accuracy ranges from 99.8% to 99.9%.

11. Conclusion:

From the above methods and results we can see that random forest has the highest accuracy and the cross-validation value is greater for this model. From bootstrap random forest gives us the best prediction result for the dataset.

12. Author Contributions:

Task	Task assigned to
Data Cleaning	Srividya
Visualization	Srividya
Pairwise association	Srividya
Supervised Learning Methods: • Logistic Regression • Gaussian Naive Bayes • K – Nearest Neighbor • Support Vector Classifier • Decision Tree • Random Forest	Madhuri
Cross Validation	Anudeep
Predictor Contributors • Variable importance	Anudeep
Estimation Stability via bootstrap	Anudeep

Group Members:

- Madhuri Vadyala (UBID – 50442120)
- Srividya Amireddy (UBID – 50469095)
- Anudeep Balagam (UBID – 50442091)

Git Repository:

https://github.com/Madhuri2198/SDMProject