CENSUS INCOME DATA ANALYSIS

05-05-2019

Akash Gupta

Table of Contents

1 BACKGROUND 2

2 OBJECTIVE 2

2.1 DATASET 3
2.2 TECHNIQUES USED 5

3 EXPLORATORY DATA ANALYSIS & DATA MANIPULATION: 5

3.1 INITIAL DATA LOADING AND DATA MANIPULATION: 5
3.2 EXPLORATORY DATA ANALYSIS 6
3.2.1 Geographical distribution of people across different countries in the dataset 6
3.2.2 Distribution of Income 7
3.2.3 Correlation between different numerical variables using correlation matrix 7
3.2.4 Distribution of Education 8
3.2.5 Distribution of Gender & Race 10
3.2.6 Distribution of Age 11
3.2.7 Pairplot between weeks worked per year & Age 12
3.2.8 Stacked Bar Graph of Race & Gender vs Income 12
3.2.9 Economic Inequalities between male & female 13

4 MODEL EVALUATION 17

4.1 LOGISTIC REGRESSION 18
4.2 K- NEAREST NEIGHBOURS 21
4.3 LINEAR DISCRIMINANT ANALYSIS 24
4.4 RANDOM FOREST 26
4.5 DECISION TREE 28

5 MODEL COMPARISON 30

6 CONCLUSION 31

7 HOW ANALYSED DATA CAN BE USED ? 31

1 Background

The prominent inequality of income based on gender is a huge concern especially in the United States. The principle of universal moral equality ensures sustainable development and improve the economic stability of a nation. Governments in different countries have been trying their best to address this problem and provide an optimal solution. This study aims to show the usage of machine learning and data mining techniques in providing a solution to the income equality problem. The UCI Adult Dataset has been used for the purpose. Classification has been done to predict whether a person's yearly income in US falls in the income category of either greater than 50K Dollars or less equal to 50K Dollars category based on a certain set of attributes.

2 Objective

The overall objective of the project is to determine whether the income level of a person exceeds over 50K per year, how income is controlled by race, age and sex and how the above factors discriminate female's role in employment. This has been achieved by performing extensive exploratory data analysis on the Adult Census dataset taken from UCI Machine Learning repository and finding some interesting relations and insights from the data. We applied various classification and regression techniques which we learned in BUAN6340 class which included Logistic regression, KNN, Linear Discriminant Analysis, Random Forest, Decision Tree models and compared the results of the techniques to find the best model.

2.1 Dataset

A. Data overview

The dataset used in this project has 199,523 records and a binomial label indicating a salary of <50K or >50K USD. 94% of the records in the dataset have a class label of <50K. The data has been divided into a training set containing 133,680 records and a test dataset containing 65,843 records.

There are 40 attributes consisting of thirty three nominal and seven continuous attributes. The employment class describes the type of employer such as self-employed or federal and occupation describes the employment type such as farming, clerical or managerial. Education contains the highest level of education attained such as high school or doctorate. The relationship attribute has categories such as unmarried or husband and marital status has categories such as married or separated. The other nominal attributes are country of residence, gender and race. The continuous attributes are age, hours worked per week, education number (numeric representation of the education attribute), capital gain and loss.

B. Attributes

Response Variable - Income: <=50K (94%), >=50K (6%)

AAGE	Age classification is based on the age of the person at his/her last birthday. The adult universe(i.e., population of marriageable age) is comprised of persons 15 years old and over for March supplement data and for CPS labor force data.
ACLSWKR - class of worker	This refers to the broad classification of the person's employer. On the March file, these broad classifications for current jobs are private, government, self-employed, without pay, and never worked. Private and government workers are considered "wage and salary workers;" this classification scheme includes self-employed, incorporated persons in with "private" workers. For the longest job held last year, this class of worker scheme includes private; government by level/Federal, State, and local; self-employed incorporated, self-employed unincorporated or farm; and without pay. The wage and salary category for longest job held includes private, government (all levels), and self-employed incorporated.
ADTIND	Industry code where the person worked

ADTOCC	Occupation code of the person
AGI	Adjusted gross income
AHGA	Highest education level
AHRSPAY	Wage per hour
AHSCOL	Enrolled in educational institute last week
AMARITL	Marital status
AMJIND	Major industry code where the person worked
AMJOCC	Major occupation code of the person
ARACE	Race of the person
AREORGN	Hispanic origin
ASEX	Sex/Gender
AUNMEM	Member of a labor union
AUNTYPE	Reason for unemployment
AWKSTAT	Full or part time employment status
CAPGAIN	Capital gains in dollars
CAPLOSS	Capital Losses
DIVVAL	Dividends from stocks
FEDTAX	Federal income tax liability
FILESTAT	Tax filer status
GRINREG	Region of previous residence
GRINST	State of previous residence
HHDFMX	Detailed household and family status
HHDREL	Detailed household summary in household
MARSUPWT	March supplement final weight
MIGMTR1	Migration code change in MSA
MIGMTR3	Migration code change in REG
MIGMTR4	Migration code move within REG
MIGSAME	Lived in this house 1 year ago
MIGSUN	Migration previous res in sunbelt
NOEMP	Number of persons worked for employer
PARENT	Family members under 18
PEARNVAL	Total personal earnings
PEFNTVTY	Country of birth for father
PEMNTVTY	Country of birth for mother
PENATVTY	Country of birth self
PRCITSHP	Citizenship
PTOTVAL	Total personal income
SEOTR	Own business or self employed
TAXINC	Taxable income amount
VETQVA	Fill questionnaire for veteran's admin
VETYN	Veterans benefits
WKSWORK	Number of weeks worked in year

2.2 Techniques Used

We have used almost all the techniques which we have learned in the BUAN6340 class which includes correlation matrix to find the correlation matrix between the numerical variables in the dataset, extensive exploratory data analysis is carried out with various plots like bar plots, correlation plots, strip plot, pairs plot etc. Various regression techniques like Logistic regression, KNN, Linear Discriminant Analysis, Random Forest, Decision Tree models and compared the results of the techniques to find the best model. Then a comparison is drawn based on the confusion matrix and accuracy of the model and the best model is then chosen.

3 Exploratory Data Analysis & Data Manipulation:

3.1 Initial data loading and data manipulation:

Columns	FATHER	MOTHER	SELF	GRINST
No. missing values	6713	6119	3393	708

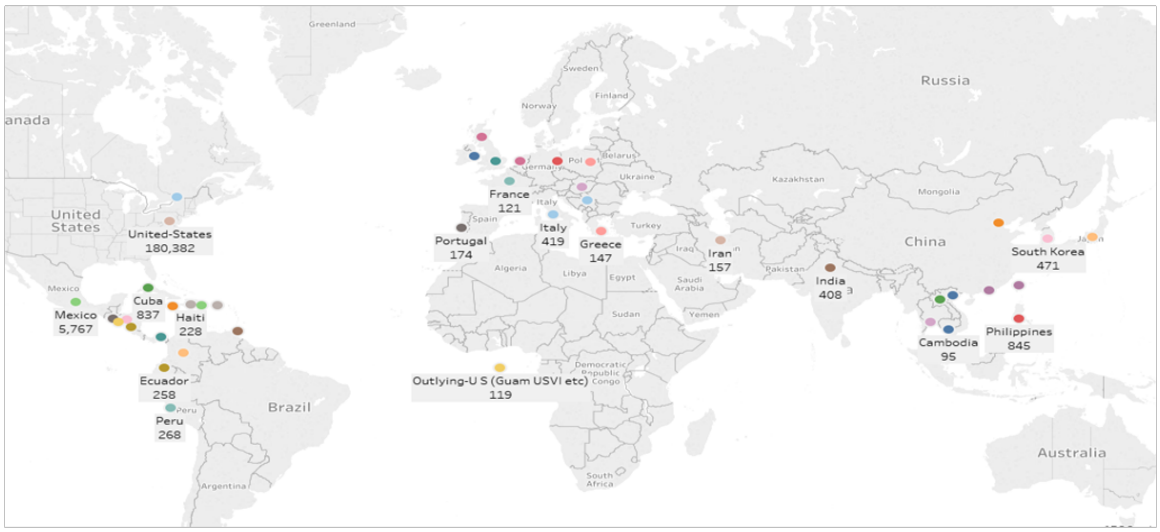
- We removed following columns from the dataset - 'MIGMTR1', 'MIGMTR3', 'MIGMTR4','MIGSUN','*(pred)','AGI','AHRSPAY'
- These columns had more than 95 thousand missing values per column
- Removing records corresponding to these columns would result in lesser number of dataset observations.
- Imputing these many records with mean, mode etc. would make data biased.
- Predicting these many records would make final income predictions unreliable
- Imputed mode values for above columns with categorical data
- Imputed mode values in place of irrelevant values already present in the data.

- Created dummy variables for all categorical fields for classification results.
- Split the data into training and testing for model evaluation

```
In [20]: X_train, X_test, Y_train, Y_test=train_test_split( X, Y, test_size=0.33,random_state=23)

In [21]: X_train.shape
         X_test.shape

Out[21]: (133680, 462)

Out[21]: (65843, 462)
```

3.2 Exploratory Data Analysis

3.2.1 Geographical distribution of people across different countries in the dataset



The above map shows population distribution of people in the dataset across the world. As we can see, about 180,000 of people are from the United States and the rest 20,000 people have migrated to the US and hence have been included in the dataset with migration variables which we further dropped.

3.2.2 Distribution of Income

The Pie Chart above depicts the distribution of income across people in the dataset. We can see that about 94% of the people in the dataset have income less than 50K and 6% of people in the dataset have income greater than 50K

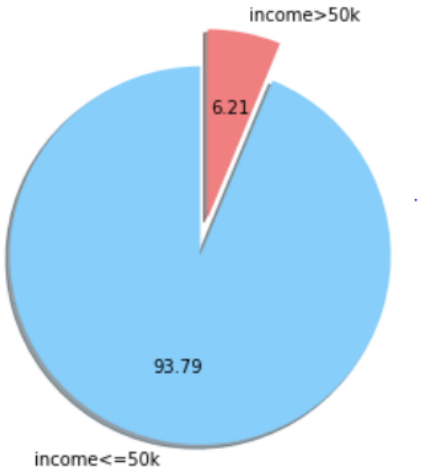


Figure 3.2.1 Pie Chart showing Income Distribution

3.2.3 Correlation between different numerical variables using correlation matrix

The Correlation is the association between two or more variables. The correlation matrix is used to investigate the dependence between multiple variables at the same time. The result is a table containing the correlation coefficients between each variable and the others.

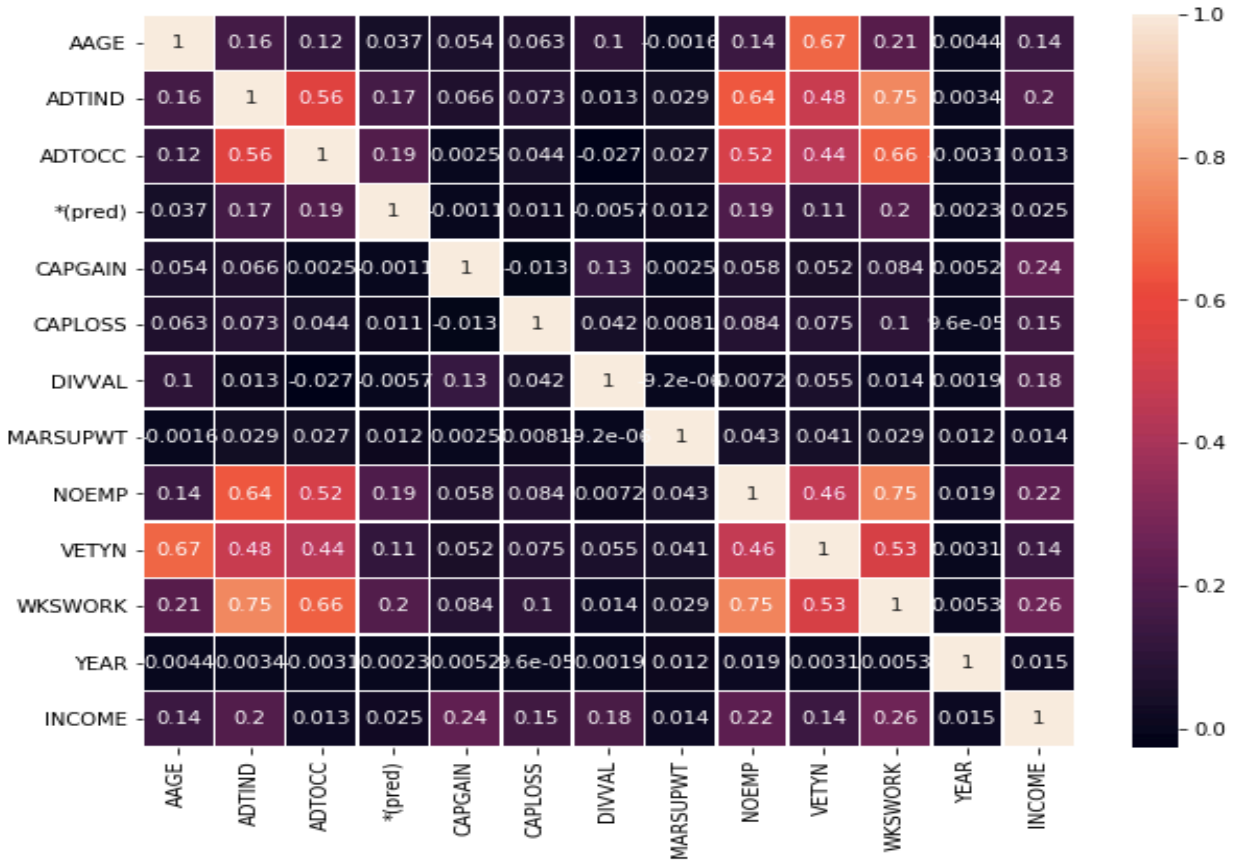


Figure 3.2.3 Correlation matrix using heat map

Each cell in the table shows the correlation between two variables. From the heat map we can see that not many variables are correlated except weeks worked in year (WKSWORK) with number of persons worked for employer (NOEMP) and Industry code (ADTIND)

3.2.4 Distribution of Education

The bar graph below shows that majority of people are high school graduates. There are almost 48k high school graduates in the dataset.

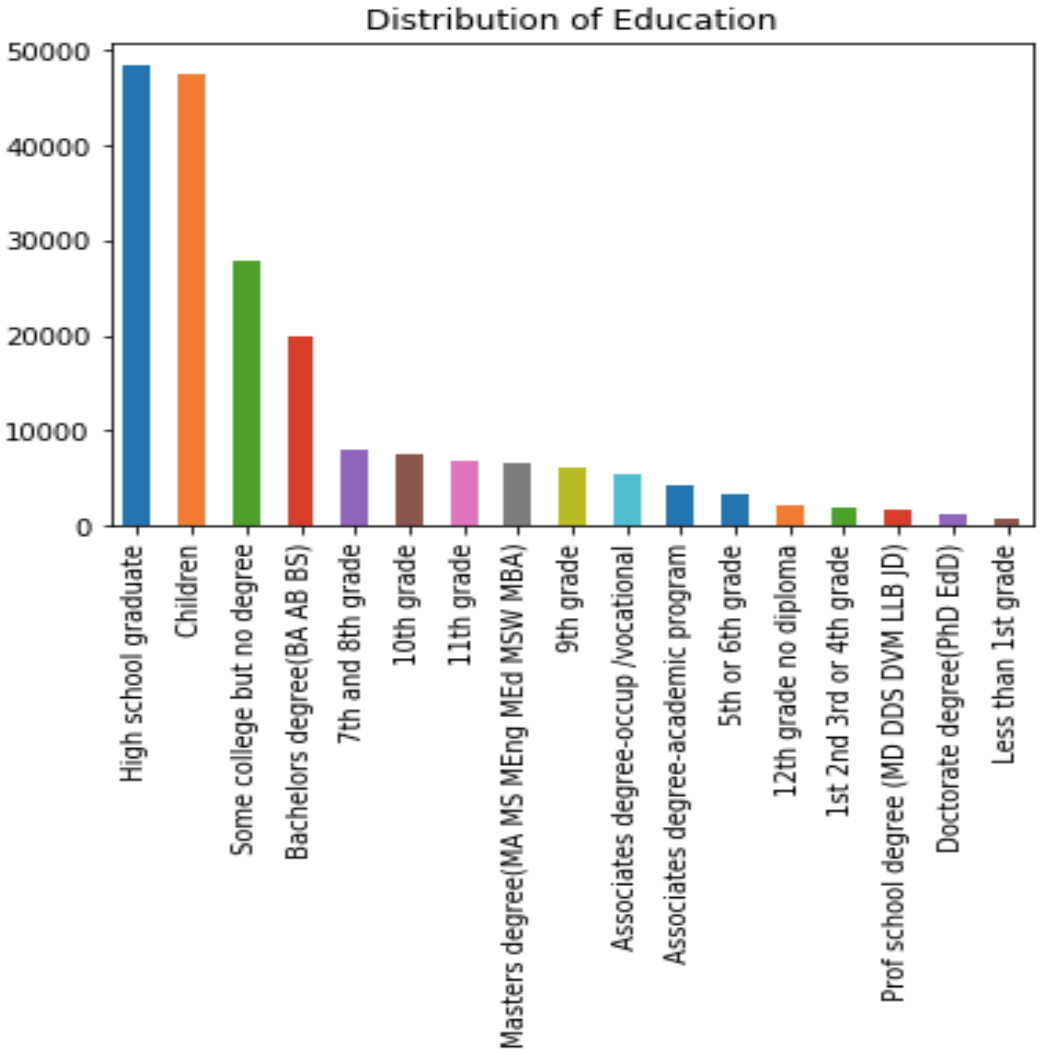


Figure 3.2.4 Distribution of Education

3.2.5 Distribution of Gender & Race

The bar graph shows the distribution of gender & race. Most of the people are whites and the distribution of gender is almost equal between females and males

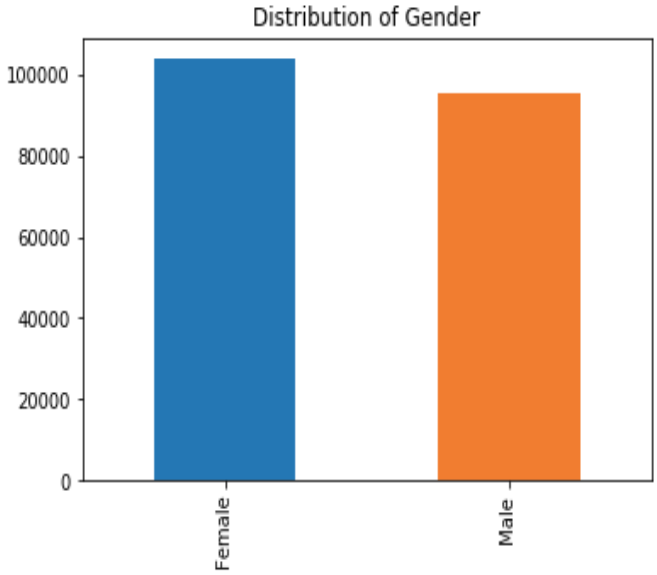


Figure 3.2.5 Distribution of Gender

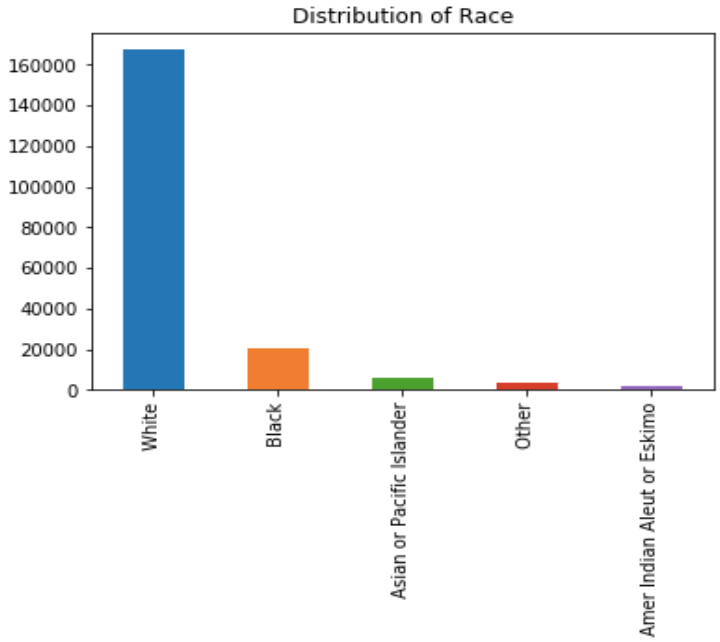


Figure 3.2.5 Distribution of Race

3.2.6 Distribution of Age

The below graphs show distribution of age. From the boxplot we can infer that 50 percentage of the people are between age 18 – 50 years. The median age in the dataset is around 36

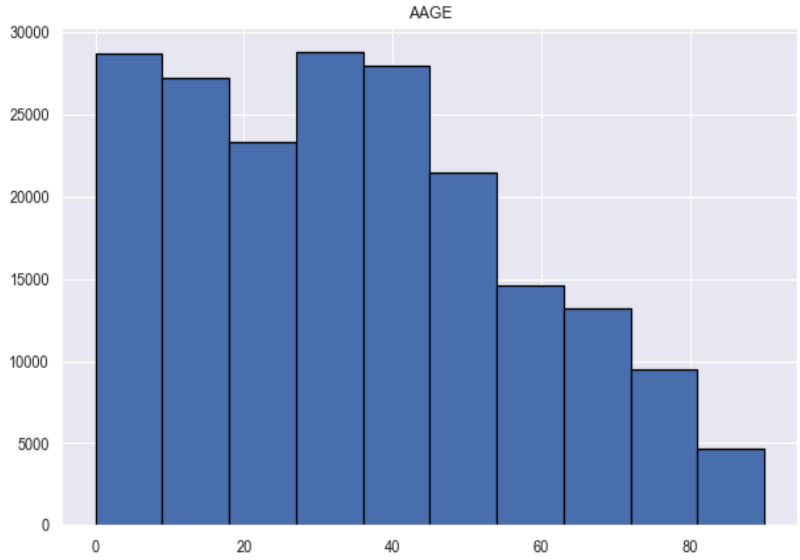


Figure 3.2.6 Histogram of Distribution of Age

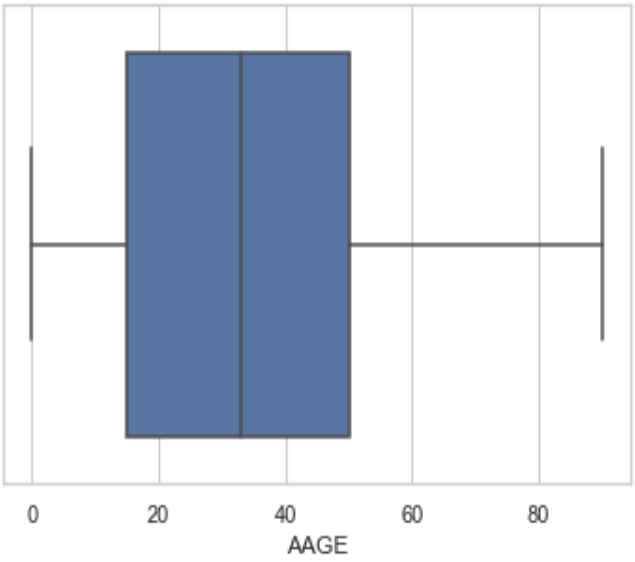


Figure 3.2.6 Boxlot showing Distribution of Age

3.2.7 Pairplot between weeks worked per year & Age

The pairs plot builds on two basic figures, the density and the scatter plot. The density on the diagonal allows us to see the distribution of a single variable while the scatter plots on the upper and lower triangles show the relationship between two variables. For example, the left-most plot in the second row shows the scatter plot of weeks worked per year versus age.

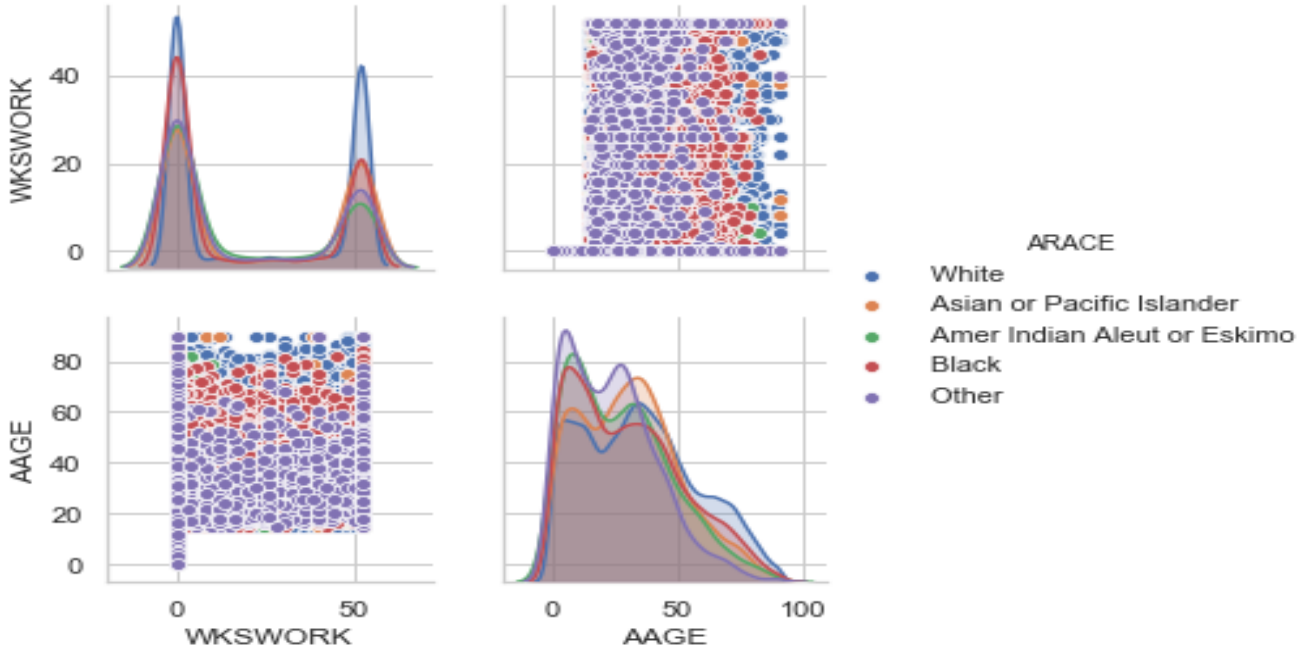


Figure 3.2.6 Pairplots showing Distribution of Weeks worked per year & Age

3.2.8 Stacked Bar Graph of Race & Gender vs Income

This is an example of creating stacked bar graphs which shows the income based on gender & race. From the graph we can see that males earn more than 50k $ per annum whereas most of the females earn<50K $ per annum.

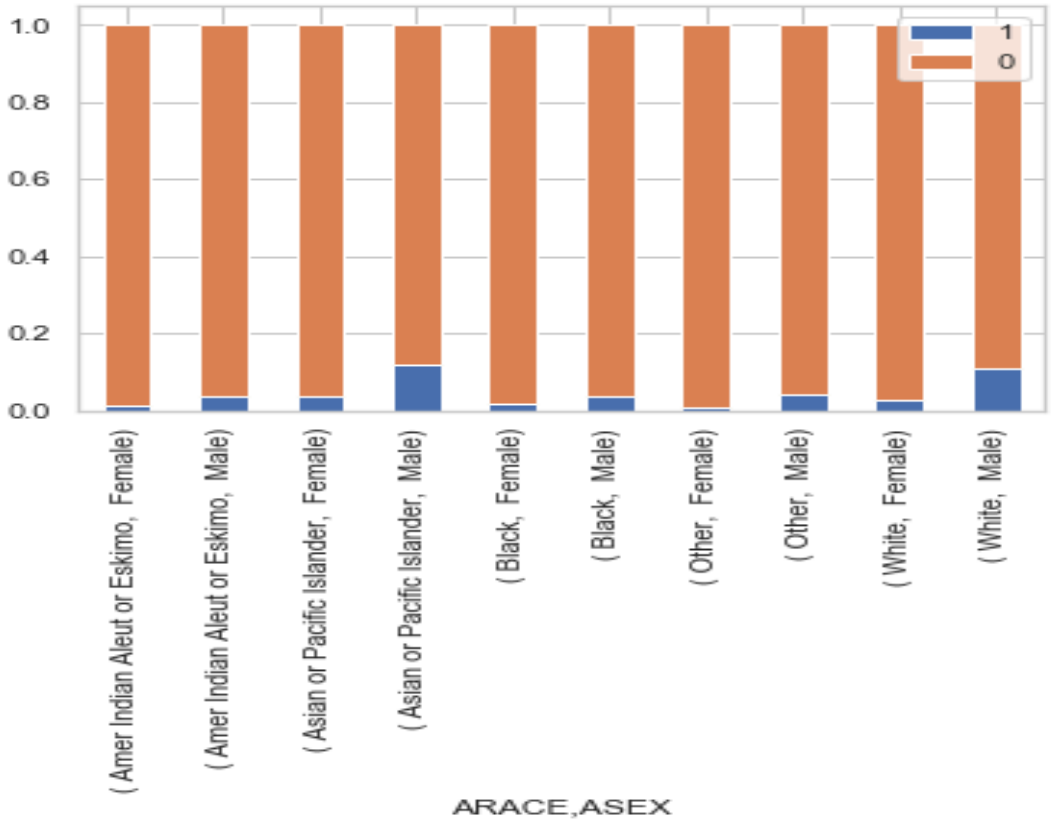


Figure 3.2.6 Stacked Bar for Race and Gender Versus Income

3.2.9 Economic Inequalities between male & female

In this entry we focus on visualizing data on economic inequalities between men and women during the years 1994 – 1995 with concise explanations.

a. The Graph displaying income division across Education Levels

The graph shows that if a person has a Bachelor's Degree, then there is a high probability of earning income more than 50k

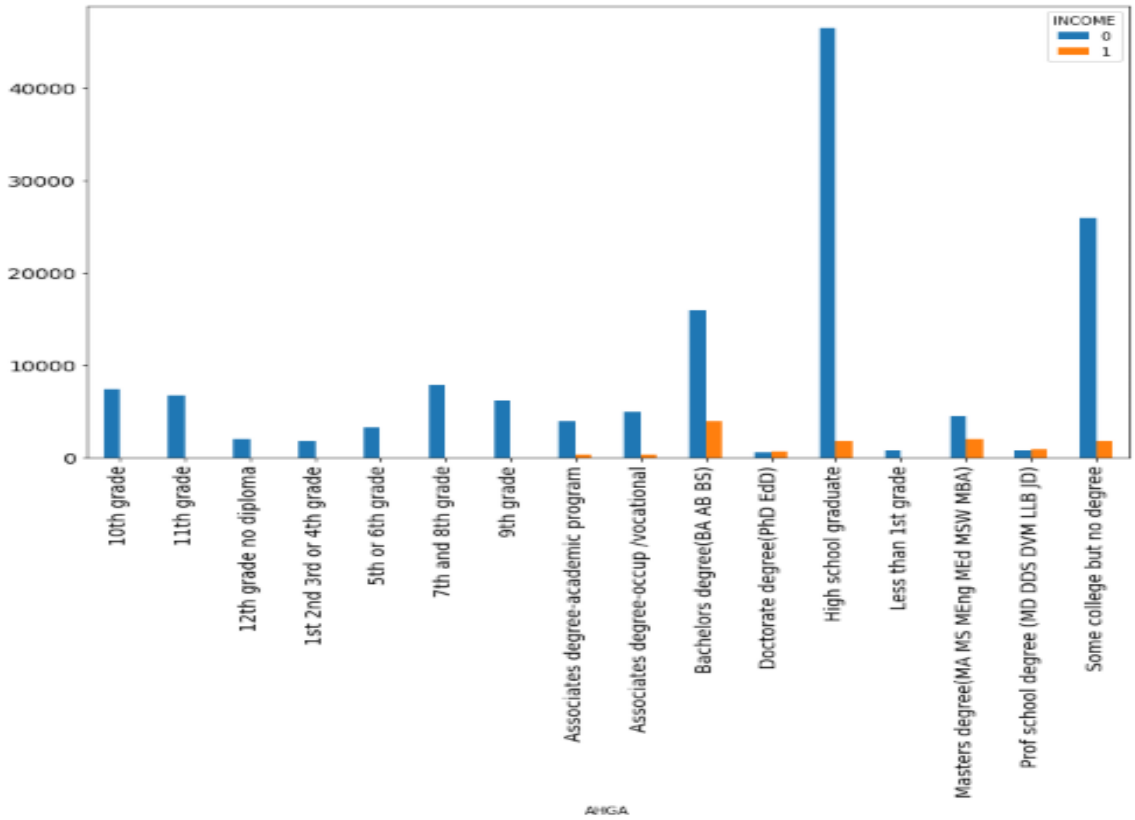


Figure 3.2.7 (a) Graph displaying Income across education levels

b. Graph displaying the Distribution of Male and Female Across the education levels

The below plot gives us a clear picture on how the gender is distributed across school level education to doctorate level. We notice that women and men are distributed equally across certain categories like 1st,2nd and 3rd Grades, 12th grade diploma, bachelor's degree

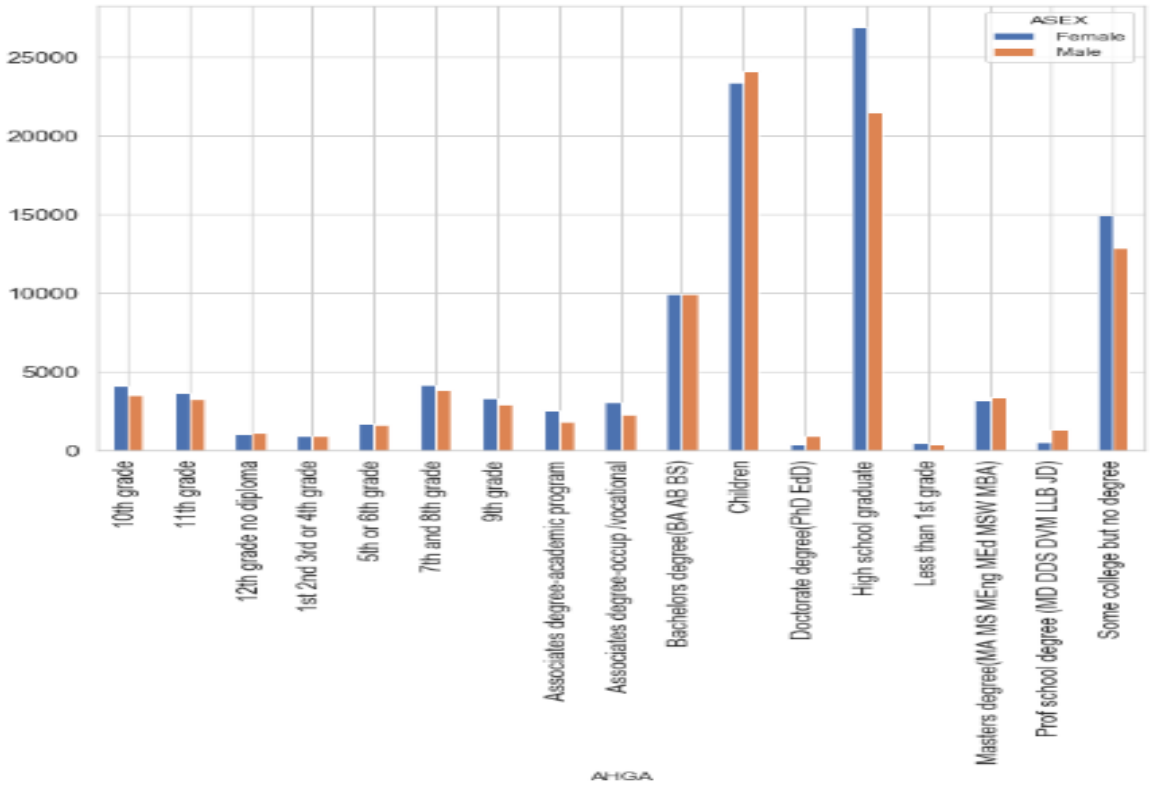


Figure 3.2.7 (b) Graph displaying Gender across education levels

c. Graph displaying Income Distribution based on Gender

The bar graph shows distribution of number of people who earn income between male and female and we notice that male has an upper hand with income> 50k.

From the above graph, we noticed that women and men are in equal numbers across various education levels and we still find a difference in wages which justifies the economic inequality by gender

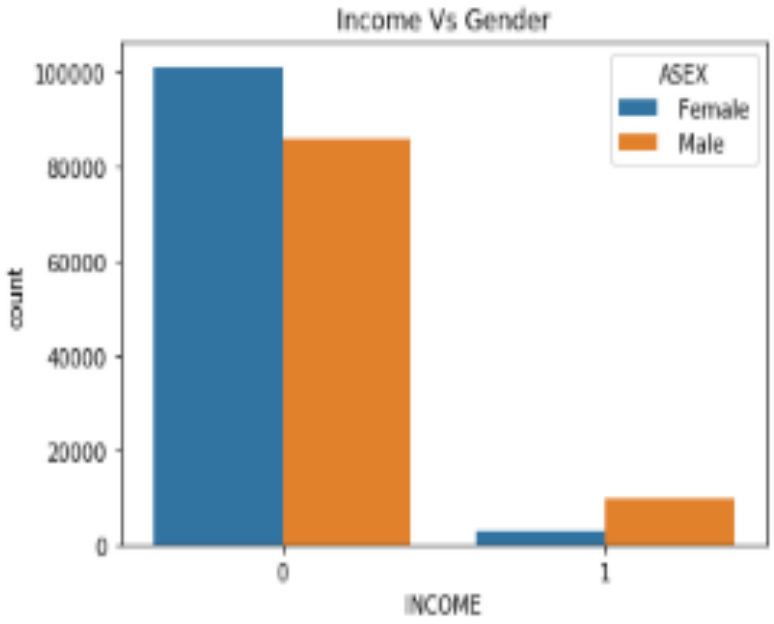


Figure 3.2.7 (c) Graph displaying Income distribution based on gender

d. Plot displaying how income is controlled by age and gender

There is a greater distribution of income>50k among males between age 35 – 60 and there is a moderate distribution of income>50k among females between age 40 – 60 which clearly shows that gender pay gap is larger for older workers

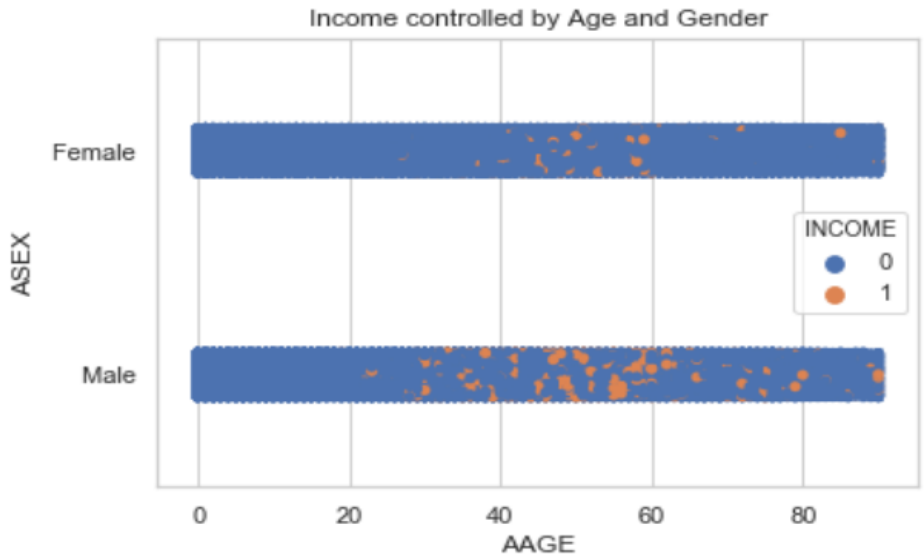


Figure 3.2.7 (d) Graph displaying Income across age and gender

4 Model Evaluation

- **Logistic Regression**
- **K Nearest Neighbors**
- **Linear Discriminant Analysis**
- **Random Forest**
- **Decision Trees**

4.1 Logistic Regression

After the cleaning the data and splitting the data into training and testing we used the training data to run a logistic regression on our model. Since our dependent variable is Binary and we are predicting the income whether it is greater than 50k or not. We are able to classify our data based on various features and using these features we are able to predict the income of the population in the data whether they a=earn more than 50k per year or less than that.

We used sklearn package in order to run our logistic model

```
In [24]: from sklearn.linear_model import LogisticRegression
         regressor=LogisticRegression()

In [25]: regressor.fit(X_train, Y_train.values.ravel())

         C:\Users\suraj\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
         to 'lbfgs' in 0.22. Specify a solver to silence this warning.
           FutureWarning)

Out[25]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='warn',
                   n_jobs=None, penalty='l2', random_state=None, solver='warn',
                   tol=0.0001, verbose=0, warm_start=False)
```

We check for regression coefficients, their shape and also the intercept values in order to derive meaningful insights.

```
In [26]: regressor.coef_

Out[26]: array([[-9.71108141e-03, -4.57582910e-02,  1.01035356e-04,
                  6.07210841e-04,  1.82552811e-04,  4.57498648e-05,
                  1.70106247e-01, -2.52214041e-02,  2.80561999e-01,
                  3.29477334e-02, -6.65162090e-02, -2.09846921e-02,
                 -2.42319631e-02, -2.52490582e-02, -2.56888471e-02,
                 -2.50268131e-02, -2.51979232e-02, -2.39800156e-02,
                 -2.48485546e-02, -2.50022045e-02, -2.45742283e-02,
                 -2.36880827e-02, -2.44813890e-02, -2.31769873e-02,
                 -2.43385899e-02, -2.30433183e-02, -5.99760878e-02,
                 -2.10889348e-02, -7.98291471e-02, -1.57551589e-01,
                 -1.36854004e-01, -1.30766602e-01, -3.48076583e-01,
                 -2.89153011e-01, -7.45618085e-01, -7.99076444e-01,
                 -8.67106474e-01, -9.10805978e-01, -9.39920515e-01,
                 -6.89167554e-01, -7.63524322e-01, -4.30242302e-01,
                 -2.37007770e-01, -2.78562817e-01, -2.19536748e-01,
                 -4.99777685e-02,  3.40998716e-02,  5.41921926e-03,
                  1.56446375e-01,  1.35901048e-01,  3.40678524e-02,
                  8.09212906e-02,  1.11041648e-01,  2.11443565e-01,
                  1.22862190e-01,  1.30637862e-01,  2.32600984e-01,
```

Then we created predictions which will store the predicted values of our testing set.

```
In [37]: from sklearn.metrics import classification_report
         print(classification_report(Y_test,y_pred_logistic_test))
```

```
                  precision    recall  f1-score   support

               0       0.96      0.99      0.97     61739
               1       0.71      0.39      0.50      4104

       micro avg       0.95      0.95      0.95     65843
       macro avg       0.83      0.69      0.74     65843
    weighted avg       0.94      0.95      0.95     65843
```

The precision is the ratio tp / (tp + fp) where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

The recall is the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

The F-beta score can be interpreted as a weighted harmonic mean of the precision and recall, where an F-beta score reaches its best value at 1 and worst score at 0.

The F-beta score weights recall more than precision by a factor of beta. beta == 1.0 means recall and precision are equally important.

The support is the number of occurrences of each class in y_true.

We can see the confusion matrix which represents the misclassifications as well as correctly classified observation count. In order to make this confusion matrix informative we created our own confusion matrix by creating a dataframe.

[33]:
```
pd.DataFrame(
    confusion_matrix(Y_test, y_pred_logistic_test),
    columns=['Predicted Income not>50k', 'Predicted Income>50k'],
    index=['True Income not>50k', 'True Income>50k']
)
```

[33]:

	Predicted Income not>50k	Predicted Income>50k
True Income not>50k	61081	658
True Income>50k	2513	1591

Here are the training and testing accuracies of our Logistic Regression model –

In [31]:
```
print("Logistic regression accuracy for training data:",accuracy_score(Y_train, y_pred_logistic_train))
```
```
Logistic regression accuracy for training data: 0.952939856373429
```

In [32]:
```
print("Logistic regression accuracy for testing data:",accuracy_score(Y_test, y_pred_logistic_test))
```
```
Logistic regression accuracy for testing data: 0.9518399829898395
```

To further evaluate our model, we use AKAIKE INFORMATION CRITERION and BAYESIAN INFORMATION CRITERION.

Here are the AIC and BIC values of our Logistic Regression model for both training and testing data –

In [91]:
```
resid = np.array([i[0] for i in Y_train.values.tolist()]) - np.array(y_pred_logistic_train.tolist())
sse = sum(resid**2)
k = 462
n = 65843

AIC = (2*k) - (2*np.log(sse))
BIC = (n*np.log(sse/n)) + (k*np.log(n))

print("AIC of regression model:",AIC)
print("BIC of regression model:",BIC)
```
```
AIC of regression model: 906.5062493608599
BIC of regression model: -149483.54010942433
```

The lower the AIC & BIC values the better is the model and therefore we can say our model stands good for now and we will compare these values when we run more models to get the best possible model.

```
In [90]: resid = np.array([i[0] for i in Y_test.values.tolist()]) - np.array(y_pred_logistic_test.tolist())
         sse = sum(resid**2)
         k = 462
         n = 65843

         AIC = (2*k) - (2*np.log(sse))
         BIC = (n*np.log(sse/n)) + (k*np.log(n))

         print("AIC of regression model:",AIC)
         print("BIC of regression model:",BIC)

            AIC of regression model: 907.8763954509233
            BIC of regression model: -194590.80461344542
```

4.2 K- Nearest Neighbours

KNN is a non-parametric algorithm, which means that it does not make any assumptions on the underlying data distribution. KNN is a type of lazy learning where the function is only approximated locally and all computation is feferred until classification.

It is based on feature similarity, which means how closely out-of-sample features resemble our training set, determines how we classify a given data point.

Why did we choose k=5 as we used hit and trial method to get the best possible results from our K-nearest neighbours model.

Small K values gave us higher influence on the results
Large K-values will make it computationally expensive.

```
In [35]: from sklearn.neighbors import KNeighborsClassifier

In [37]: from sklearn.neighbors import KNeighborsClassifier
         knn = KNeighborsClassifier(n_neighbors=5)
         knn.fit(X_train,Y_train)
         y_pred = knn.predict(X_train)

         /anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:3: DataConversionWarning: A column-vector y was passed w
         hen a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
           This is separate from the ipykernel package so we can avoid doing imports until

Out[37]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                    weights='uniform')
```

Here are the training and testing accuracies of our KNN model –

```
In [36]: #accuracy of knn
         print("KNN accuracy for training data:",accuracy_score(Y_train, y_pred_knn_train))

            KNN accuracy for training data: 0.9536729503291442

In [38]: print("KNN accuracy for testing data:",accuracy_score(Y_test, y_pred_knn_test))

            KNN accuracy for testing data: 0.9397658065397992
```

The Test set accuracy came out to be lesser than the training accuracy as the test set data was never seen before and we can see that the test set results are significant.

Here is the confusion matrix of our KNN model -

```
In [39]: pd.DataFrame(
             confusion_matrix(Y_test, y_pred_knn_test),
             columns=['Predicted Income not>50k', 'Predicted Income>50k'],
             index=['True Income not>50k', 'True Income>50k']
         )
```

Out[39]:

	Predicted Income not>50k	Predicted Income>50k
True Income not>50k	60781	958
True Income>50k	3008	1096

We can see that the overall error rate is less as there are very few misclassifications which leads to lesser overall error rate. We check for sensitivity and specificity values for knn model and compare it with other models.

Here are the AIC and BIC values of our KNN model for both training and testing data –

In [42]:
```
resid = np.array([i[0] for i in Y_train.values.tolist()]) - np.array(y_pred_knn_train.tolist())
sse = sum(resid**2)
k = 462
n = 65843

AIC = (2*k) - (2*np.log(sse))
BIC = (n*np.log(sse/n)) + (k*np.log(n))

print("AIC of KNN model:",AIC)
print("BIC of KNN model:",BIC)
```

```
AIC of KNN model: 906.5376501981239
BIC of KNN model: -150517.30277340903
```

The lower the AIC & BIC values the better is the model and therefore we can say our model stands good for now and we will compare these values when we run more models to get the best possible model.

In [92]:
```
resid = np.array([i[0] for i in Y_test.values.tolist()]) - np.array(y_pred_knn_test.tolist())
sse = sum(resid**2)
k = 462
n = 65843

AIC = (2*k) - (2*np.log(sse))
BIC = (n*np.log(sse/n)) + (k*np.log(n))

print("AIC of KNN model:",AIC)
print("BIC of KNN model:",BIC)
```

```
AIC of regression model: 907.4289733818405
BIC of regression model: -179860.9989661364
```

4.3 Linear Discriminant Analysis

LDA is a very common technique used for supervised classification problems. LDA is a dimension reduction technique used as a preprocessing step in Machine Learning and pattern classification applications.

```
In [40]: from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
         lda = LDA(n_components=1)

         X_train1 = lda.fit_transform(X_train, Y_train)
         X_test1 = lda.transform(X_test)
         y_pred_lda_train = lda.predict(X_train)

C:\Users\suraj\Anaconda3\lib\site-packages\sklearn\utils\validation.py:761: DataConversionWarning: A column-vector y was passe
d when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\suraj\Anaconda3\lib\site-packages\sklearn\discriminant_analysis.py:388: UserWarning: Variables are collinear.
  warnings.warn("Variables are collinear.")
```

Here are the training and testing accuracies of our LDA model –

```
In [43]: #accuracy of lda
         print("LDA accuracy for training data:",accuracy_score(Y_train, y_pred_lda_train))

LDA accuracy for training data: 0.9448234590065829

In [44]: print("LDA accuracy for testing data:",accuracy_score(Y_test, y_pred_lda_test))

LDA accuracy for testing data: 0.9443524748264812
```

The Test set accuracy came out very close to the training accuracy and we can see that the test set results are significant.

Here is the confusion matrix of our LDA model –

```
In [45]: pd.DataFrame(
             confusion_matrix(Y_test, y_pred_lda_test),
             columns=['Predicted Income not>50k', 'Predicted Income>50k'],
             index=['True Income not>50k', 'True Income>50k']
         )
```

Out[45]:

	Predicted Income not>50k	Predicted Income>50k
True Income not>50k	60052	1687
True Income>50k	1977	2127

We can see that the overall error rate is less as there are very few misclassifications which leads to lesser overall error rate .We check for sensitivity and specificity values for LDA model and compare it with other models.

Here are the AIC and BIC values of our LDA model for both training and testing data –

In [49]:
```
resid = np.array([i[0] for i in Y_train.values.tolist()]) - np.array(y_pred_lda_train.tolist())
sse = sum(resid**2)
k = 462
n = 65843

AIC = (2*k) - (2*np.log(sse))
BIC = (n*np.log(sse/n)) + (k*np.log(n))

print("AIC of LDA model:",AIC)
print("BIC of LDA model:",BIC)
```

```
AIC of LDA model: 906.1880264695271
BIC of LDA model: -139007.16519241157
```

The lower the AIC & BIC values the better is the model and therefore we can say our model stands good for now and we will compare these values when we run more models to get the best possible model.

In [93]:
```
resid = np.array([i[0] for i in Y_test.values.tolist()]) - np.array(y_pred_lda_test.tolist())
sse = sum(resid**2)
k = 462
n = 65843

AIC = (2*k) - (2*np.log(sse))
BIC = (n*np.log(sse/n)) + (k*np.log(n))

print("AIC of LDA model:",AIC)
print("BIC of LDA model:",BIC)
```

```
AIC of regression model: 907.587378548412
BIC of regression model: -185075.93465741808
```

4.4 Random Forest

Random Forest or Random Decision Forest is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

```
In [46]: from sklearn.ensemble import RandomForestClassifier
         rfc = RandomForestClassifier(max_depth=2, random_state=0)

         rfc.fit(X_train1, Y_train)
         y_pred_rfc_train = rfc.predict(X_train1)

         C:\Users\suraj\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The default value of n_estimators wi
         ll change from 10 in version 0.20 to 100 in 0.22.
           "10 in version 0.20 to 100 in 0.22.", FutureWarning)
         C:\Users\suraj\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: DataConversionWarning: A column-vector y was passed when a
         1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
           This is separate from the ipykernel package so we can avoid doing imports until

Out[46]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                     max_depth=2, max_features='auto', max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
                     oob_score=False, random_state=0, verbose=0, warm_start=False)
```

Here are the training and testing accuracies of our Random Forest model –

```
In [53]: #accuracy of RFC
         print("RFC accuracy for training data:",accuracy_score(Y_train, y_pred_rfc_train))

            RFC accuracy for training data: 0.9499925194494315

In [54]: print("RFC accuracy for testing data:",accuracy_score(Y_test, y_pred_rfc_test))

            RFC accuracy for testing data: 0.9490758319031636
```

The Test set accuracy came out very close to the training accuracy and we can see that the test set results are significant.

Here is the confusion matrix of our Random Forest model –

In [52]:
```
pd.DataFrame(
    confusion_matrix(Y_test, y_pred_rfc_test),
    columns=['Predicted Income not>50k', 'Predicted Income>50k'],
    index=['True Income not>50k', 'True Income>50k']
)
```

Out[52]:

	Predicted Income not>50k	Predicted Income>50k
True Income not>50k	61191	548
True Income>50k	2805	1299

Here are the AIC and BIC values of our Random Forest model for both training and testing data -

In [56]:
```
resid = np.array([i[0] for i in Y_train.values.tolist()]) - np.array(y_pred_rfc_train.tolist())
sse = sum(resid**2)
k = 462
n = 65843

AIC = (2*k) - (2*np.log(sse))
BIC = (n*np.log(sse/n)) + (k*np.log(n))

print("AIC of random forest model:",AIC)
print("BIC of random forest model:",BIC)
```

```
AIC of random forest model: 906.384757020928
BIC of random forest model: -145483.83004035207
```

The lower the AIC & BIC values the better is the model and therefore we can say our model stands good for now and we will compare these values when we run more models to get the best possible model.

In [94]:
```
resid = np.array([i[0] for i in Y_test.values.tolist()]) - np.array(y_pred_rfc_test.tolist())
sse = sum(resid**2)
k = 462
n = 65843

AIC = (2*k) - (2*np.log(sse))
BIC = (n*np.log(sse/n)) + (k*np.log(n))

print("AIC of random forest model:",AIC)
print("BIC of random forest model:",BIC)
```

```
AIC of regression model: 907.7647785070675
BIC of regression model: -190916.2073962979
```

4.5 Decision Tree

A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs and utility.

```
In [53]: from sklearn.tree import DecisionTreeRegressor
         # create a regressor object
         dtr = DecisionTreeRegressor(random_state = 0)

         # fit the regressor with X and Y data
         dtr.fit(X_train,Y_train)

         y_pred_dtr_train = dtr.predict(X_train)

Out[53]: DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
                    max_leaf_nodes=None, min_impurity_decrease=0.0,
                    min_impurity_split=None, min_samples_leaf=1,
                    min_samples_split=2, min_weight_fraction_leaf=0.0,
                    presort=False, random_state=0, splitter='best')
```

Here are the training and testing accuracies of our Decision Tree model –

```
In [29]: print("Decision Tree accuracy for training data:",accuracy_score(Y_train, y_pred_dtr_train))
         Decision Tree accuracy for training data: 0.9412654320625421

In [61]: print("Decision Tree accuracy for testing data:",accuracy_score(Y_test, y_pred_dtr_test))
         Decision Tree accuracy for testing data: 0.9318530443631062
```

The Test set accuracy came out comparatively less close to the training accuracy and we can see that the test set results are significant.

Here is the confusion matrix of our Decision Tree model -

In [59]:
```
pd.DataFrame(
    confusion_matrix(Y_test, y_pred_dtr_test),
    columns=['Predicted Income not>50k', 'Predicted Income>50k'],
    index=['True Income not>50k', 'True Income>50k']
)
```

Out[59]:

	Predicted Income not>50k	Predicted Income>50k
True Income not>50k	59462	2277
True Income>50k	2210	1894

Here are the AIC and BIC values of our Decision Tree model for both training and testing data –

In [64]:
```
resid = np.array([i[0] for i in Y_train.values.tolist()]) - np.array(y_pred_dtr_train.tolist())
sse = sum(resid**2)
k = 462
n = 65843

AIC = (2*k) - (2*np.log(sse))
BIC = (n*np.log(sse/n)) + (k*np.log(n))

print("AIC of decision tree model:",AIC)
print("BIC of decision tree model:",BIC)
```

```
AIC of decision tree model: 925.3862943611199
BIC of decision tree model: -771042.9415854823
```

The lower the AIC & BIC values the better is the model and therefore we can say our model stands good for now and we will compare these values when we run more models to get the best possible model.

In [95]:
```
resid = np.array([i[0] for i in Y_test.values.tolist()]) - np.array(y_pred_dtr_test.tolist())
sse = sum(resid**2)
k = 462
n = 65843

AIC = (2*k) - (2*np.log(sse))
BIC = (n*np.log(sse/n)) + (k*np.log(n))

print("AIC of decision tree model:",AIC)
print("BIC of decision tree model:",BIC)
```

```
AIC of regression model: 907.182120788048
BIC of regression model: -171734.24129959714
```

5 Model Comparison

MODEL	ACCURACY VALUES
LOGISTIC REGRESSION	95.1839 %
K NEAREST NEIGHBORS	93.9765 %
LINEAR DISCRIMINANT ANALYSIS	94.4352 %
RANDOM FOREST	94.9075 %
DECISION TREES	93.1853 %

Table 5.1 : Accuracy Comparison

In terms of Accuracy we can see that Logistic provides the best accuracy amongst the models in **Logistic model.**

Models	AIC Values
LOGISTIC REGRESSION	907.8763
K NEAREST NEIGHBORS	907.4289
LINEAR DISCRIMINANT ANALYSIS	907.5873
RANDOM FOREST	907.7647
DECISION TREES	907.1821

Table 5.2 : AIC Comparision

Here we compare the AIC values and we can say that **Decision Trees** provides the lowest AIC values among all models

Model	Sensitivity	Specificity
LOGISTIC REGRESSION	96.05%	70.74 %
K NEAREST NEIGHBOR	95.28%	53.36 %
LINEAR DISCRIMINANT ANALYSIS	96.81%	55.77 %
RANDOM FOREST	95.62%	70.33 %
DECISION TREES	96.42%	45.41 %

Table 5.3 : Sensitivity & Specificity Comparison

Sensitivity (also called the true positive rate, the recall, or probability of detection in some fields) measures the proportion of actual positives that are correctly identified as such.

Specificity (also called the true negative rate) measures the proportion of actual negatives that are correctly identified as such .

We calculate the sensitivity and specificity of the models in order to check which has the best optimum values and we found that Logistic Regression would be best compared to other models and the overall analysis.

6 Conclusion

As far as AIC values are concerned, we see that there is not much difference within any of the models, hence we take other parameters like accuracy, sensitivity, specificity into consideration & we can certainly say that logistic model is better fit to predict the income of an individual based on the census data.

7 How Analysed Data Can Be Used ?

- Spreading awareness among schools and colleges to maintain gender equality during on-campus placements
- We can target the 6% population with income > 50k for fundraising and starting campaigns for women empowerment
- Establish structures, policies, objectives in every organization to ensure gender balance
- This analysis is useful for loan lenders to know the american salary distribution and target relevant customers and segment them appropriately