How to Get Rich!

Victoria Routon
Samantha Wolownik
Aaron Portal

Abstract

In this project, we want to identify the variables that will indicate an individual's income level. Our group is motivated by this topic as we are all taking the next steps towards careers and are interested in what variables drive monetary success. We are using a census dataset that contains specific information about each person such as age, marital status, occupation as well as their education level. We apply several algorithms such as Multivariable Linear Regression, Logistic Regression, Naïve Bayes, Random Forest, K-means and Hierarchical clustering algorithm to determine which attributes have a significant impact on different income levels among adults and which do not. We include several different visualization tools for each algorithm to better visualize the results we find.

Introduction

Knowing what variables affect someone's income is something we believe many people would want to know. And although there are multiple known factors that will affect one's income such as, but not limited to, education, gender, age, region, experience, ect., our goal was to get a closer look at how much these variables, and others in the dataset, actually affect one's income. Some questions we might be asking ourselves are 'How much of an effect does age have on income?', 'We know there is an income gap based on gender, but is it also affected by other aspects as well?', and 'Based on our data, what trait has the greatest effect on one's income?'. We are hopeful that we will find results that answer these questions, but we are also aware that our data might not support the questions we are asking. Furthermore, during our exploratory analysis, we might come across new questions and ways to look at the data we have not thought of before.

Our data set comes from a 1994 census. Thus, we should keep in mind that this data was collected before the 2008 financial crisis and other events that might've affected the economy, thus may not accurately represent to the same degree the effect of these variables on income now, especially during these difficult times. However, we hope to find some interesting insight from our analysis that could show a more general trend in how these attributes affect one's income. Furthermore, even without a to date dataset or analysis, we might also be able to compare our results with what we know of how income is determined in today's climate. We could look at things like the change in the gender pay gap over time, or the average level of education for a certain income group. We also understand that our data might be limited due to the fact that the income attribute consists of only two different variables: below $50K a year and at or above $50K a year. But this type of classification might be useful in determining a more general idea of what income is expected based on certain attributes. Furthermore, due to inflation, the value of $50,000 has changed, and is roughly worth just below $90,000 today. Thus, the baseline income in our data, is not an accurate representation of what income to expect. With all other factors remaining the same, keep in mind when reading our results that the $50K separator between incomes is much higher today.

Our plan is to perform multiple regression and classification algorithms in order to determine which ones are best at analysing and representing the data provided, while answering the questions we presented above. We will be doing this by using the Python language to implement our desired algorithms and finding ways to visualize the data and our results. What follows is our implementation of our chosen algorithms and the results they yield.

Data Exploratory (EDA)

Data description

The dataset that was used to apply the numerous machine learning algorithms consisted of 15 different variables, with 14 feature attributes and 1 target variable. The data regarded census data of a large population generalised under a weight and distributed under the data.

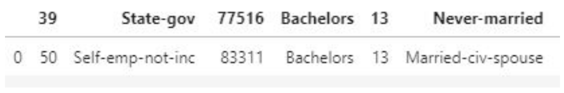
- Age
    - The numerical age of the entry
    - A Continuous variable
    - Greater than or equal to 0 that has the Integer datatype
- Workclass
    - The employment status of the entry in question
    - Categorical variable
    - Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked
- FNLWGT
    - The final weight or the number of people the entry refers to in the data
    - Continuous variable
    - Integer datatype
- Education
    - The highest level of education achieved
    - Categorical
    - Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool
    - Object<String> datatype
- Education-Num
    - The highest level of education achieved mapped to a numerical value
    - Continuous variable
    - Integer datatype
- Marital-Status
    - The current marital status
    - Categorical variable
    - Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse
    - Object<String> datatype
- Occupation
    - The current job of the entry
    - Categorical variable
    - Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces
    - Object<String> datatype
- Relationship
    - The relationship of the entry relative to their family
    - Categorical variable

  - Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried
  - Object<String> datatype
- Race
  - The race of the entry
  - Categorical variable
  - White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black
  - Object<String> datatype
- Sex
  - The binary sex of the entry
  - Categorical variable
  - Female, Male
- Capital-Gain
  - The individual's net positive capital gain
  - Continuous variable
  - Integer datatype
- Capital-Loss
  - The individual's net capital loss
  - Continuous variable
  - Integer datatype
- Hours-Per-Week
  - The numerical number of hours the entry works a week
  - Continuous variable
  - Integer datatype
- Native-Country
  - The country of origin of the individual
  - Categorical variable
  - United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands
  - Object<String> datatype
- Income
  - If the individual makes more or less than $50K
  - Target variable
  - Categorical variable
  - >50K, <=50K
  - Object<String> datatype
  - Number of Values
    - 24719 <=50k
    - 7841 >50k

Analyzing the data and manually observing the features, we determined that the attribute "fnlwgt" could be removed due to its unproportional values compared to any of the other variables as well as its relevance to the target feature. The "fnlwgt" represented the weight of individuals that correlated with a series of the other attributes in the dataset. So essentially it was a numerically encoded representation of how the data was represented using the other variables. We saw that it was hampering specifically the implementation of clustering and how the results could be interpreted, and so collectively agreed to remove the "fnlwgt" from the dataset.

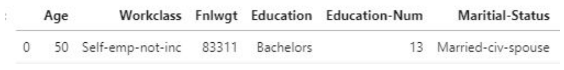
Preprocessing

We applied several preprocessing steps to the data before we attempted to do calculations and perform machine learning algorithms on the data. Initially we began by printing out the data, where we discovered that the dataset was missing the column names. This was easily remedied by explicitly defining the column names for the data.



	39	State-gov	77516	Bachelors	13	Never-married
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse

Figure 1 – Before defining column names



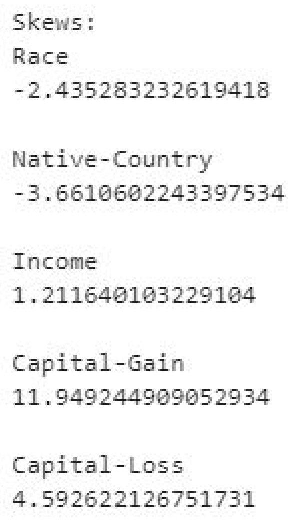
	Age	Workclass	Fnlwgt	Education	Education-Num	Maritial-Status
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse

Figure 2 – After defining column names

After this, the data was analyzed for null values, of which none were discovered. Next duplicate values were checked for, and it was found that there were 24 rows of duplicate data in the set. After the duplicate rows were identified, they were promptly dropped from the dataset and the count of rows was checked to verify that they had been removed.

Next we removed any outliers from the dataset. This was accomplished by determining the Interquartile Range (IQR) and then using the IQR rule to isolate the outliers from the data and then remove them from the dataset.

After the outliers were removed, we checked for skewed attributes and after setting a threshold of 1 and discovered several features that were skewed. This was done by running the the features through the skew function from Pandas and comparing it to a skew value threshold. The features “Race”, ”Native-Country”, ”Income”, “Capital-Gain”, ”Capital-Loss” were discovered to have higher skews. Negative values corresponded to left-skews, while positive values correlated to right-skews.



```
Skews:
Race
-2.435283232619418

Native-Country
-3.6610602243397534

Income
1.211640103229104

Capital-Gain
11.949244909052934

Capital-Loss
4.592622126751731
```

Figure 3 – Left and right skewed variables

Once the skew were discovered the square root method was applied several of the features to see how it was affected, but was ultimately not included as it would have to be applied to all the features in the dataset, not just the skewed variables, and there were only 5 different skewed variables. Looking at the variables, however, we discovered that our target feature was actually skewed itself.

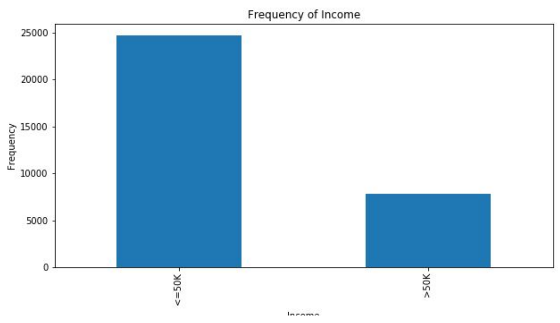
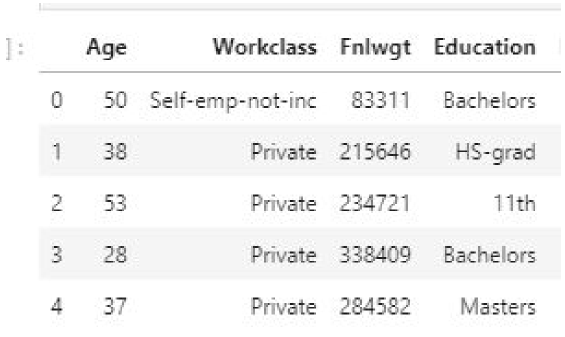


Figure 4 – Skew/Frequency of target variable “Income”

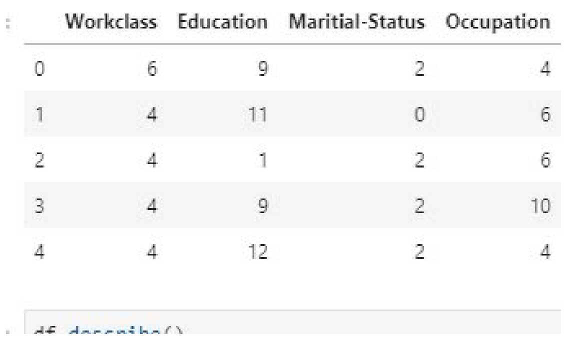
After discussing the skewed results of the target variable, we determined that we should leave it as it was. We agreed that the data was collected from the census and thus was real data, and so should be left with the skew intact since it represented the actual distribution of incomes above and below the $50,000 threshold.

Following this we used a label encoder to convert the categorial format of the data into numerical data so that it could be applied to certain algorithms, such as Multiple Linear Regression. The “obj64” string variables were isolated to their own separate data frame from the “int64” features that were already numerical values. Then the string variables were encoded by a label encoder into numerical values in a dataframe. Once the data was encoded the new encoded data frame was combined with the original numerical features into a completely numerically encoded data set consisting of all the original features.



	Age	Workclass	Fnlwgt	Education
0	50	Self-emp-not-inc	83311	Bachelors
1	38	Private	215646	HS-grad
2	53	Private	234721	11th
3	28	Private	338409	Bachelors
4	37	Private	284582	Masters

Figure 3 – Data before label encoding



	Workclass	Education	Maritial-Status	Occupation
0	6	9	2	4
1	4	11	0	6
2	4	1	2	6
3	4	9	2	10
4	4	12	2	4

Figure 5 – Data after label encoding

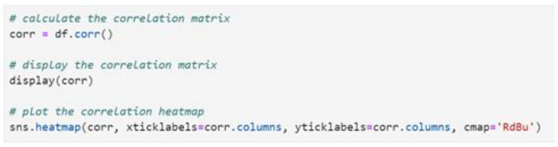
After the data was correctly formatted with the necessary variables removed and the data encoded to numerical values, the new preprocessed dataset could be applied to the machine learning algorithms and results could be attained.

Algorithms

Multivariable Linear Regression

The first algorithm that we decided to apply to our dataset was multivariable linear regression. This approach is very similar to linear regression with the difference that now we want to consider several explanatory variables to predict the target variable. However, before we do so, we need to keep in mind about some of the assumptions that the model needs to meet in order to produce the correct output. The first assumption is that there shouldn’t be any multicollinearity in our data. The reason why we don’t want to see multicollinearity is that if we have two variables that behave similar and have the same impact on the model it’s hard to separate what is the impact that these two variables have on the model. We also want to add variables to the model that could tell us something new and useful to our model and having the variables that say the same thing would create redundancy and hence affect the model’s performance.

To check whether our dataset passes the multicollinearity test, we plot a heap map to better visualize the results (Figure 1). On the diagonal we expect to see a very dark blues that's going to be correlation of one because the correlation of a particular column to itself is just 1, but other than that we don't notice anything disturbing that could affect the multicollinearity, therefore we can say that our dataset passed the multicollinearity test.



```
# calculate the correlation matrix
corr = df.corr()

# display the correlation matrix
display(corr)

# plot the correlation heatmap
sns.heatmap(corr, xticklabels=corr.columns, yticklabels=corr.columns, cmap='RdBu')
```

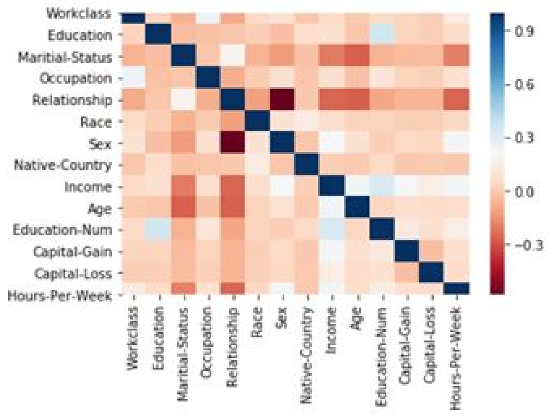
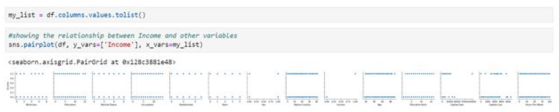


Figure 1 – correlation matrix

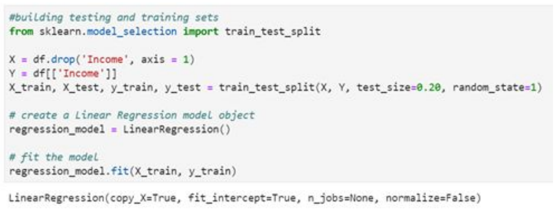
Another important assumption for multivariable regression is that the data needs to have a linear relationship between the dependent and independent variables. We visualize it by plotting the pair plot using a seaborn library where we compare the target variable with the rest of the variables (Figure 2). As we can see, there is no linear relationship between them, therefore we conclude that our dataset is not suitable for this type of algorithm and hence will not produce an accurate result.



```
my_list = df.columns.values.tolist()

#showing the relationship between Income and other variables
sns.pairplot(df, y_vars=['Income'], x_vars=my_list)

<seaborn.axisgrid.PairGrid at 0x128c3881e48>
```

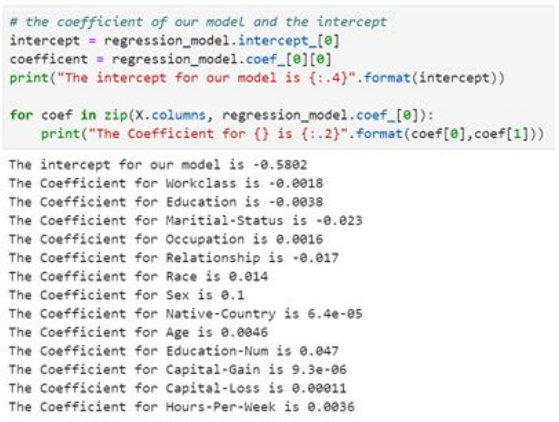
Figure 2 – relationship between the independent and dependent variables

For the testing purposes, I decided to apply the model regardless to prove that this type of dataset would not work for this algorithm. After I split the data into a training and testing set, I fit the training set into the model (Figure 3) and I found the intercept which is the value of the dependent variable when all the independent variables are equal to zero as well as the coefficients (Figure 4). For each slope coefficient is it the estimated change in the dependent variable for a one-unit change in that particular independent variable, holding the other independent variables constant.



```
#building testing and training sets
from sklearn.model_selection import train_test_split

X = df.drop('Income', axis = 1)
Y = df[['Income']]
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.20, random_state=1)

# create a Linear Regression model object
regression_model = LinearRegression()

# fit the model
regression_model.fit(X_train, y_train)

LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

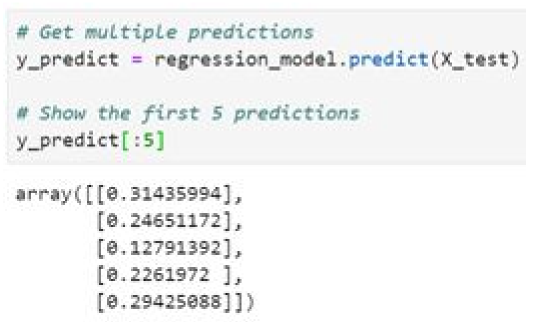
Figure 3 – fitting the model



```
# the coefficient of our model and the intercept
intercept = regression_model.intercept_[0]
coefficent = regression_model.coef_[0][0]
print("The intercept for our model is {:.4}".format(intercept))

for coef in zip(X.columns, regression_model.coef_[0]):
    print("The Coefficient for {} is {:.2}".format(coef[0],coef[1]))

The intercept for our model is -0.5802
The Coefficient for Workclass is -0.0018
The Coefficient for Education is -0.0038
The Coefficient for Maritial-Status is -0.023
The Coefficient for Occupation is 0.0016
The Coefficient for Relationship is -0.017
The Coefficient for Race is 0.014
The Coefficient for Sex is 0.1
The Coefficient for Native-Country is 6.4e-05
The Coefficient for Age is 0.0046
The Coefficient for Education-Num is 0.047
The Coefficient for Capital-Gain is 9.3e-06
The Coefficient for Capital-Loss is 0.00011
The Coefficient for Hours-Per-Week is 0.0036
```

Figure 4 – coefficient of the model and the intercept

I made the predictions and displayed the first five of them (Figure 5).

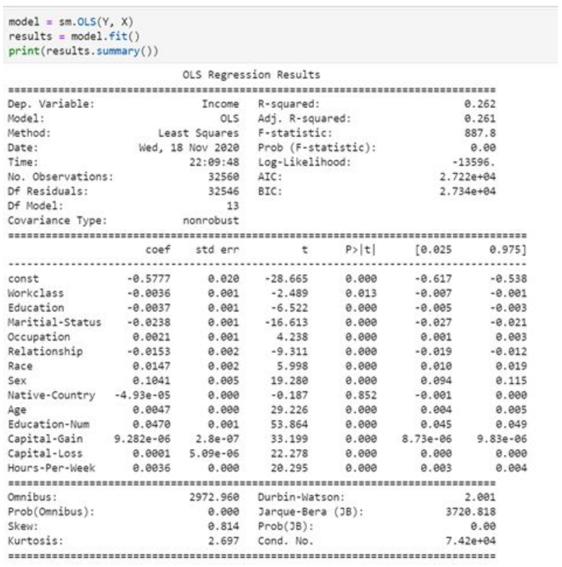


```
# Get multiple predictions
y_predict = regression_model.predict(X_test)

# Show the first 5 predictions
y_predict[:5]
```

```
array([[0.31435994],
       [0.24651172],
       [0.12791392],
       [0.2261972 ],
       [0.29425088]])
```

Figure 5 – predictions

Then we began evaluating the model by using the statsmodel to create an ordinary least square regression model and we fit our data (Figure 6). We notice that our R^2 value is pretty low, which means that our linear regression model does not fit our data very well and since adjusted R^2 penalizes us for the features that don't have significance to our model tells us that there are some features in our model that are affecting our overall model performance – in fact if we look at this table we notice that the Native-Country has a p value of 0.98 which means that it's not relevant to our model and should potentially be removed.

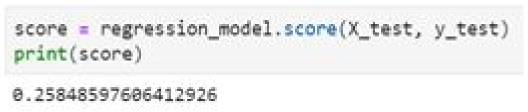


```
model = sm.OLS(Y, X)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 Income   R-squared:                       0.262
Model:                            OLS   Adj. R-squared:                  0.261
Method:                 Least Squares   F-statistic:                     887.8
Date:                Wed, 18 Nov 2020   Prob (F-statistic):               0.00
Time:                        22:09:48   Log-Likelihood:                -13596.
No. Observations:               32560   AIC:                         2.722e+04
Df Residuals:                   32546   BIC:                         2.734e+04
Df Model:                          13
Covariance Type:            nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
const             -0.5777      0.020    -28.665      0.000      -0.617      -0.538
Workclass         -0.0036      0.001     -2.489      0.013      -0.007      -0.001
Education         -0.0037      0.001     -6.522      0.000      -0.005      -0.003
Maritial-Status   -0.0238      0.001    -16.613      0.000      -0.027      -0.021
Occupation         0.0021      0.001      4.238      0.000       0.001       0.003
Relationship      -0.0153      0.002     -9.311      0.000      -0.019      -0.012
Race               0.0147      0.002      5.998      0.000       0.010       0.019
Sex                0.1041      0.005     19.280      0.000       0.094       0.115
Native-Country  -4.93e-05      0.000     -0.187      0.852      -0.001       0.000
Age                0.0047      0.000     29.226      0.000       0.004       0.005
Education-Num      0.0470      0.001     53.864      0.000       0.045       0.049
Capital-Gain    9.282e-06    2.8e-07     33.199      0.000    8.73e-06    9.83e-06
Capital-Loss       0.0001   5.09e-06     22.278      0.000       0.000       0.000
Hours-Per-Week     0.0036      0.000     20.295      0.000       0.003       0.004
==============================================================================
Omnibus:                     2972.960   Durbin-Watson:                   2.001
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3720.818
Skew:                           0.814   Prob(JB):                         0.00
Kurtosis:                       2.697   Cond. No.                     7.42e+04
==============================================================================
```

Figure 6 – OLS regression results

As we mentioned before, this dataset does not pass the linear relationship assumption that this algorithm must meet in order to produce correct output, hence we're expecting the accuracy of this model to be low (Figure 7).



```
score = regression_model.score(X_test, y_test)
print(score)
```

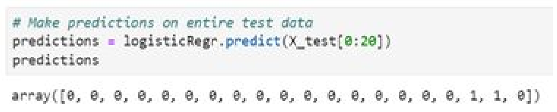
```
0.25848597606412926
```

Figure 7 – multivariable regression accuracy score

Logistic Regression

Logistic Regression is a classification algorithm that is often used when the target output is qualitative. Since our target variable has two unique values (greater or less than 50K) we hoped that this algorithm would perform better than multivariable linear regression. Since we already know that our data does not have a linear relationship between dependent and independent variables, logistic regression seems like a much better choice as this assumption does not need to be met for a logistic regression to predict accurate results.

Since we already preprocessed and split the data into training and testing sets, we created a logistic regression model, fit the data and made predictions (Figure 1).



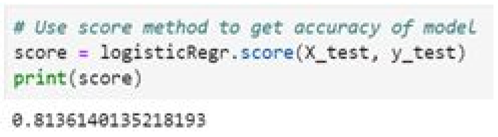
```
# Make predictions on entire test data
predictions = logisticRegr.predict(X_test[0:20])
predictions
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0])
```

Figure 1 – Logistic Regression predictions

As we can see in Figure 1, most of the 20 predictions that we made had value of 0, which is equivalent to an income of less than 50K. Very few people had a projected salary of more than 50K, in fact in the predictions that we made, only two of 20 people had income higher than 50K.

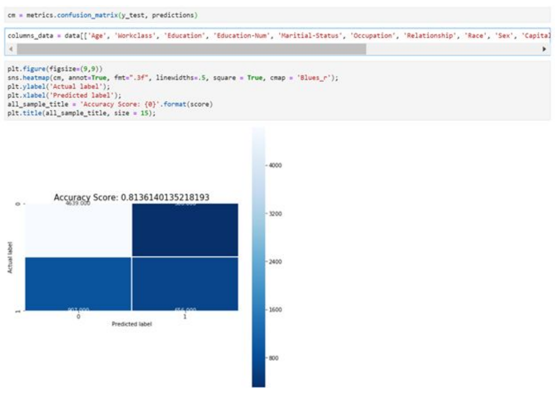
To see how accurate this algorithm was we calculate the accuracy score, which tells us how accurate our predictions are. The score is at 81% which is much better than the score we got for multivariable linear regression which was at 25% (Figure 2).



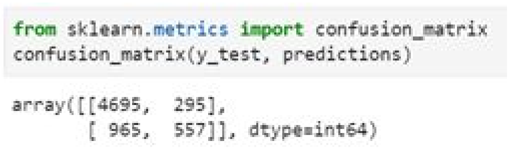
```
# Use score method to get accuracy of model
score = logisticRegr.score(X_test, y_test)
print(score)

0.8136140135218193
```

Figure 2 – logistic regression accuracy score

Next, we also want to see the confusion matrix, which helps us visualize the performance of the algorithm. So as we can see we got around 5413 of them right, however we also notice we got quite some of the false positive and false negative values which is not very good and we usually aim for these to be low, however overall this algorithm performed much better than multivariable linear regression (Figure 3).



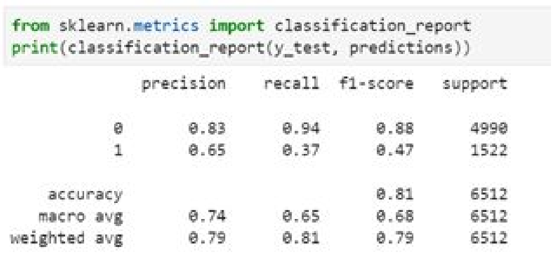
```
cm = metrics.confusion_matrix(y_test, predictions)

columns_data = data[['Age', 'Workclass', 'Education', 'Education-Num', 'Maritial-Status', 'Occupation', 'Relationship', 'Race', 'Sex', 'Capital

plt.figure(figsize=(9,9))
sns.heatmap(cm, annot=True, fmt=".3f", linewidths=.5, square = True, cmap = 'Blues_r');
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
all_sample_title = 'Accuracy Score: {0}'.format(score)
plt.title(all_sample_title, size = 15);
```



```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, predictions)

array([[4695,  295],
       [ 965,  557]], dtype=int64)
```

Figure 3 – confusion matrix

To further present the accuracy of the model, we imported a classification report that lets us see the precision, recall and f-score of the algorithm (Figure 4). Here we notice that out of the predicted cases 73% cases were identified correctly and we got a 55% of an f-score, which is called a harmonic mean of the precision and recall metric that measures a model's accuracy. As we notice logistic regression algorithm performed much better compared to multiple linear regression algorithm, however for better results we may need to look into other algorithms to receive better results.



```
from sklearn.metrics import classification_report
print(classification_report(y_test, predictions))

              precision    recall  f1-score   support

           0       0.83      0.94      0.88      4990
           1       0.65      0.37      0.47      1522

    accuracy                           0.81      6512
   macro avg       0.74      0.65      0.68      6512
weighted avg       0.79      0.81      0.79      6512
```

Figure 4 – classification report

Naive Bayes

The next algorithm that we implemented was the Naive Bayes Classifier based off of Baye's rule of probability. The Naive Bayes classifier assumes that all the features are independent, called class conditional independence The reason the Naive Bayes classifier assumes that each variable is conditionally independent is due to the amount of computation it can perform.

When it neglects to assume that the attributes are conditionally independent it drastically raises the amount of computation that the classifier must perform.

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Figure 1 – Baye's Rule

-P(h)-the Prior:
represents the probability of h occurring.

-P(D)-the Marginal Likelihood-
represents the probability of D occurring.

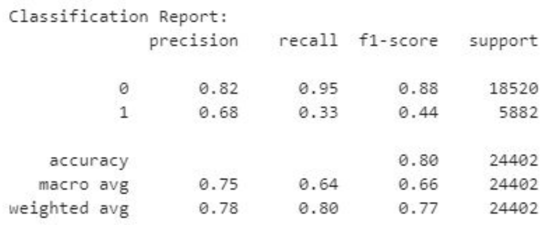
-P(h|D)-Posterior-
represents the probability of the presence of D occurring.

-P(D|h)-The Likelihood-
represents the probability of D
given the presence of D occurring.

The Naive Bayes classifier works by initially calculating the Prior probability for all the separate labels, determining the conditional probabilities for all the variables, multiplying the class conditional probability, and then multiplying the prior probability with the previous probability. Once all these steps have been completed the probability for each variable can be compared to determine the combination of variables that contains the highest probability.

Beginning the implementation of the algorithm, the first step is to format the data correctly. To simplify the computation and reduce the length of the output data we utilized the label encoded dataset to perform the Naive Bayes classifier. The base dataset was then divided into the features and the target variable and applied by the train test split to separate the data into testing and training datasets, with the training dataset being composed of 25% of the data.

After the data was successfully formatted, the algorithm could be applied to the data. The Guassian Naive Bayes model was created and fitted using the training data, and then applied to the testing data and was able to achieve an accuracy score of ~80%. Then the performance of the algorithm was analyzed using the classification report to determine the precision, recall, f1-score, and accuracy of the algorithm.



```
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.95      0.88     18520
           1       0.68      0.33      0.44      5882

    accuracy                           0.80     24402
   macro avg       0.75      0.64      0.66     24402
weighted avg       0.78      0.80      0.77     24402
```

Figure 2 - Classification Report

The precision relates to the correctly identified true results or true positives identified by the algorithm divided by the actual results from the original dataset. This provides the results that are actually relevant. The recall represents the percentage of the total relevant results that were correctly identified and classified by the algorithm. This is calculated by dividing the true positives by the predicted results from the algorithm. The F1-score is a combination of the precision and recall and can be understood as the harmonic mean between the precision and recall. As can be seen above in the Classification report the precision, recall, and f1-scores all remained relatively high ranging around 80% to 95%

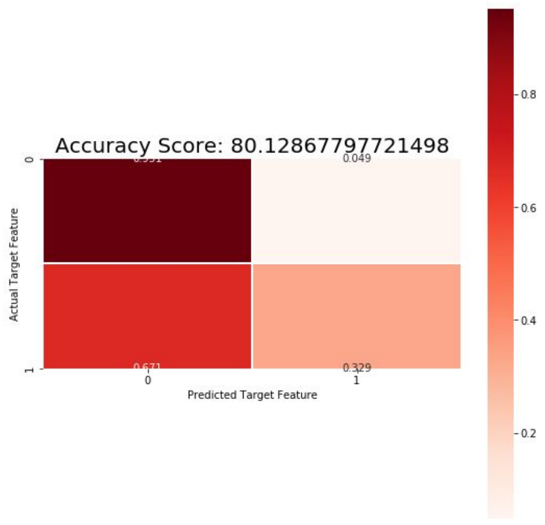
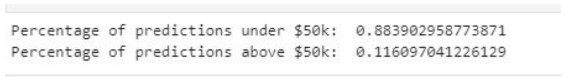


Figure 3 - Confusion Matrix

Afterwards the data was further analyzed by determining the True Positives, True Negatives, False Positives, and False Negatives from the Confusion Matrix. The true positives attained a score of ~95%, the false positives a score of ~5%, the false negatives a score of ~67%, and the false positives a score of ~33%. In regard to the high percentage of the true positives, the algorithm was highly successful. The false negatives also performed fairly well achieving an accuracy of 67%. Although not as high as the true positives it still remained in a good range. The false positives and true negatives should both have lower scores as they correlate to unwanted predictions. They both performed fairly well with the false positives reaching 5% and the true negatives hitting 33%. Considering the accuracy of the relevant correct data and the low values of the incorrect predictions it can be concluded that the Naive Bayes classifier algorithm was largely successful.

Analyzing the results of the algorithm it became apparent that the predictions made by the classifier followed suit with the original post-formatted data regarding the distribution and frequency of the target feature. The vast majority of the results produced predictions for the income level being below the $50k threshold. This was consistent with the distribution of the original data as nearly ¾ of the target feature was for individuals that had an income level less than $50k. According to the Naive Bayes classifier results, an astounding ~88% of the predictions pertained to having an income of less than $50k, while merely ~12% associated with having an income of above $50k. The results of the Naive Bayes classifier appeared to have remained consistent with the original dataset in regard to the data distribution, and as the the accuracy of the model was determined to be upwards of 80%, the algorithm was evidently very successful.



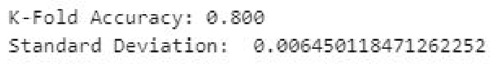
```
Percentage of predictions under $50k:  0.883902958773871
Percentage of predictions above $50k:  0.116097041226129
```

Figure 4 - Prediction Results

Random Forest

The next algorithm performed was the Random Forest model, which is an expansion of the Decision Trees classifier model. Decision trees consist of a tree graph that maps all the features based on splits that are performed on that data to achieve a prediction based on the different features that are present. Essentially to understand the results of a decision tree you simply follow the nodes that correspond to the features that are present and that you desire to determine a prediction of. The Decision Tree's leafs show the final prediction value of the combination of those features. Decision Trees are usually very easy to read and comprehend, and the paths that show the results are easy to follow. The Random Forest classifier expands the Decision Tree model by creating a forest, so to speak, of Decision Trees and comparing their results to create a final prediction metric for the algorithm. Random Trees are more accurate than a single Decision Tree as they are able to run hundreds or thousands of Decisions Trees and determine and end result using the data from the total k runs of trees.

Applying the Random Forest model, the first step was to correctly format the data. The data was converted to a numpy array and then split into testing and training sub data sets. After this the Random Forest Regressor model was created and fitted using the training data, and then applied to the testing data. The predictions were made and using the Mean Absolute Error method (MAE) an error of 20% was discovered. Further analyzing the accuracy of the model we performed the K-fold cross validation method to determine the accuracy of the algorithm and discovered a better accuracy score of 80% as well as a very low standard deviation of 0.06%.



```
K-Fold Accuracy: 0.800
Standard Deviation:  0.006450118471262252
```

Figure 1 - K-fold Accuracy

After running the accuracy measurements and determining the accuracy of the model was acceptable, we decided to graph a visual representation of the output of the Random Forest and were met with this grotesquely over-expanded Decision Tree. Although we were conscious of the skewed distribution of the target variable, we failed to realize how this would affect the Random Forest Regressors' results. Although the algorithm performed fairly well, it outputted results that were essentially unreadable and impossible to follow. Due to how the target variable was incredibly skewed as well the number of attributes in the dataset, the tree outputted a very bloated result. The more options and variables that were present only increased the size of the tree, and due to how the target variable was disrupted with a large amount of the values falling into the >50k category the results of the Random Forest were produced. Obviously these results were not the most useful and virtually impossible to analyze given the scale of a single tree from the model, and how it was almost impossible to even read the tree's predictions, much less even follow the paths of the tree.

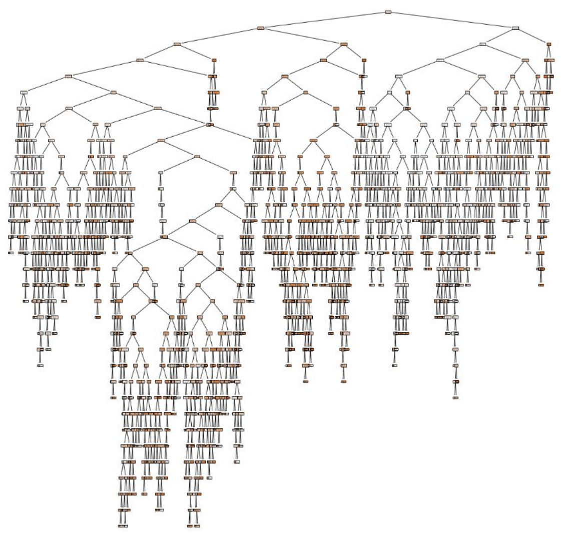


Figure 2 - Random Forest Results

Although the Random Forest model's results had reasonably high accuracies, due to the representation and scale of the predictions through the Decision Tree, the final results were unable to be applied or analyzed.

Clustering
K-Means Clustering

Finally, Clustering was performed on the model to look at any other trends that might exist within the data. More specifically, we fit a K-Means Clustering model to the data, then analyzed the results that followed.

A brief description of K-Means Clustering: K-Means Clustering is an unsupervised machine learning algorithm that is used to find groups of commonalities between data points that may not be clearly labeled. In *Figure 1*, we can see what the dataset looked like before performing K-Means Clustering. Because we wanted to find commonalities aside from Income, we created a new dataset that consists of every attribute except the Income attribute. K-Means Clustering was performed on this new data set. However, before performing the algorithm, the optimal number of clusters needed to be found. To do this we performed the elbow method, found the silhouette coefficients, and determined the cost of each number of clusters. The results of these can be seen in *Figure 2*. From these results, we found that although it was one of the most expensive number of clusters, 2 was the optimal number of clusters to go with because it creates the greatest separation between clusters. This makes sense in theory because the target (Income) of the dataset is divided into two parts: less than $50,000 a year, and at or more than $50,000 a year.

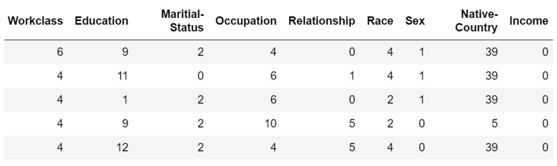
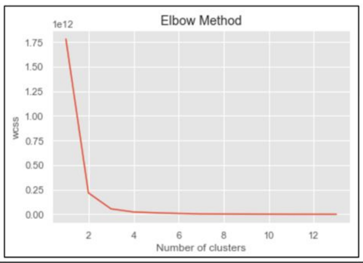
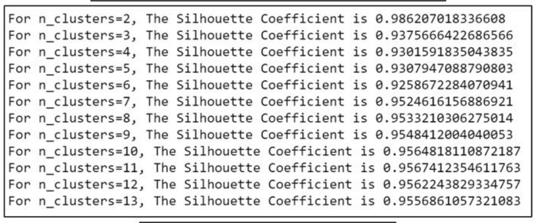


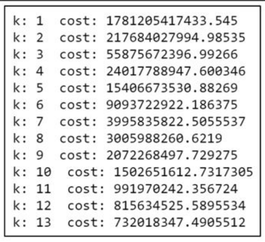
Figure 1 - Data Before Clustering

Workclass	Education	Maritial-Status	Occupation	Relationship	Race	Sex	Native-Country	Income
6	9	2	4	0	4	1	39	0
4	11	0	6	1	4	1	39	0
4	1	2	6	0	2	1	39	0
4	9	2	10	5	2	0	5	0
4	12	2	4	5	4	0	39	0





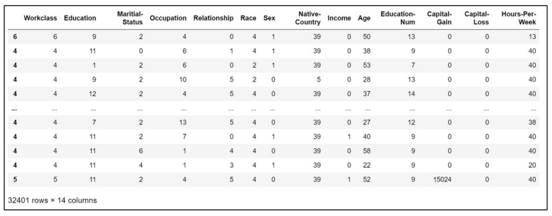
```
For n_clusters=2, The Silhouette Coefficient is 0.986207018336608
For n_clusters=3, The Silhouette Coefficient is 0.9375666422686566
For n_clusters=4, The Silhouette Coefficient is 0.9301591835043835
For n_clusters=5, The Silhouette Coefficient is 0.9307947088790803
For n_clusters=6, The Silhouette Coefficient is 0.9258672284070941
For n_clusters=7, The Silhouette Coefficient is 0.9524616156886921
For n_clusters=8, The Silhouette Coefficient is 0.9533210306275014
For n_clusters=9, The Silhouette Coefficient is 0.9548412004040053
For n_clusters=10, The Silhouette Coefficient is 0.9564818110872187
For n_clusters=11, The Silhouette Coefficient is 0.9567412354611763
For n_clusters=12, The Silhouette Coefficient is 0.9562243829334757
For n_clusters=13, The Silhouette Coefficient is 0.9556861057321083
```



```
k: 1  cost: 1781205417433.545
k: 2  cost: 217684027994.98535
k: 3  cost: 55875672396.99266
k: 4  cost: 24017788947.600346
k: 5  cost: 15406673530.88269
k: 6  cost: 9093722922.186375
k: 7  cost: 3995835822.5055537
k: 8  cost: 3005988260.6219
k: 9  cost: 2072268497.729275
k: 10  cost: 1502651612.7317305
k: 11  cost: 991970242.356724
k: 12  cost: 815634525.5895534
k: 13  cost: 732018347.4905512
```

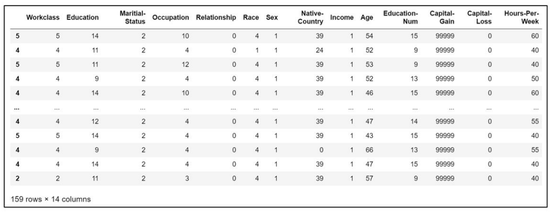
Figure 2 - In Descending Order: Elbow Method, Silhouette Coefficients, and Costs.

Now we can perform K-Means Clustering with two clusters. We fit the model to the data and created a variable label that kept track of what cluster each sample belongs to. This variable was then added to the original dataset as an attribute titled 'Cluster'. We used this Cluster attribute to split the set further into two more sets: one for the first cluster, and one for the second. We were sure to leave the Cluster attribute out of these new sets because it was redundant and useless beyond this point. *Figure 3* shows what the new datasets look like.



	Workclass	Education	Maritial-Status	Occupation	Relationship	Race	Sex	Native-Country	Income	Age	Education-Num	Capital-Gain	Capital-Loss	Hours-Per-Week
6	6	9	2	4	0	4	1	39	0	50	13	0	0	13
4	4	11	0	6	1	4	1	39	0	38	9	0	0	40
4	4	1	2	6	0	2	1	39	0	53	7	0	0	40
4	4	9	2	10	5	2	0	5	0	28	13	0	0	40
4	4	12	2	4	5	4	0	39	0	37	14	0	0	40
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4	4	7	2	13	5	4	0	39	0	27	12	0	0	38
4	4	11	2	7	0	4	1	39	1	40	9	0	0	40
4	4	11	6	1	4	4	0	39	0	58	9	0	0	40
4	4	11	4	1	3	4	1	39	0	22	9	0	0	20
5	5	11	2	4	5	4	0	39	1	52	9	15024	0	40

32401 rows × 14 columns



	Workclass	Education	Maritial-Status	Occupation	Relationship	Race	Sex	Native-Country	Income	Age	Education-Num	Capital-Gain	Capital-Loss	Hours-Per-Week
5	5	14	2	10	0	4	1	39	1	54	15	99999	0	60
4	4	11	2	4	0	1	1	24	1	52	9	99999	0	40
5	5	11	2	12	0	4	1	39	1	53	9	99999	0	40
4	4	9	2	4	0	4	1	39	1	52	13	99999	0	50
4	4	14	2	10	0	4	1	39	1	46	15	99999	0	60
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4	4	12	2	4	0	4	1	39	1	47	14	99999	0	55
5	5	14	2	4	0	4	1	39	1	43	15	99999	0	40
4	4	9	2	4	0	4	1	0	1	66	13	99999	0	55
4	4	14	2	4	0	4	1	39	1	47	15	99999	0	40
2	2	11	2	3	0	4	1	39	1	57	9	99999	0	40

159 rows × 14 columns

Figure 3 - In Descending Order: First Cluster and Second Cluster

The next step was to find the averages of all attributes in both clusters and compare them. From printing both sets of averages using the dataset.mean() function, we were able to come to the assumption that the averages in the second cluster were most likely greater than those in the first cluster. However, we wanted to prove this. Thus, we graphed the averages of each cluster against each other. We ran into a few graphical problems as the attributes ‘Capital-Gain’ and ‘Capital-Loss’ seemed to be outliers compared to the other averages. We first considered scaling the averages and replotting these scaled averages. However, we concluded that this was not necessary as capital gain and capital loss are more likely consequences of one’s income rather than causes. We also did want to calculate the difference of the means without scaling. Thus, we decided to remove these two attributes from our graphs. *Figure 4* shows the plotting of the remaining means. As we can see from the graph, more often than not, the averages of the second cluster (represented in green) are greater than the averages of the first cluster (represented in red). The only three attributes in which the average of the first cluster is greater are ‘Marital-Status’, ‘Relationship’ (which represents the professional relationship between the subject and the data collectors), and ‘Native-Country’. These might show some correlation to income, such as ‘people who are not married make less’, but the differences between these averages are so slim, it might not be too helpful. Furthermore, we are interested in the qualities that most affect a higher income. Thus, we want to focus on the positive differences of the averages.

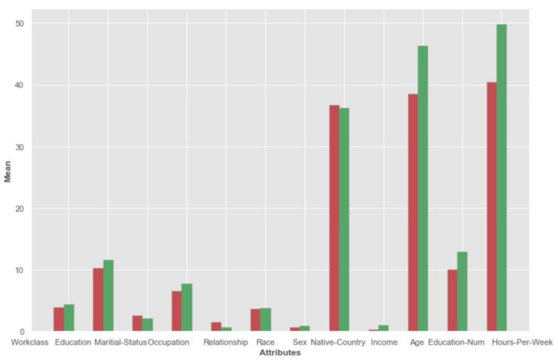


Figure 4 - Plot of Averages of the Separate Clusters Against Each other (Excluding ‘Capital-Gain’ and ‘Capital-Loss’). First Cluster in Red and Second Cluster in Green.

Figure 5 shows the plot of the averages with the positive differences.

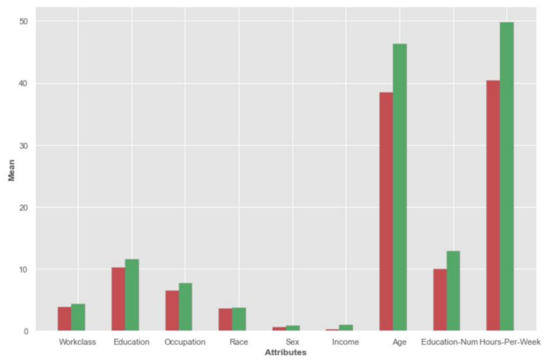
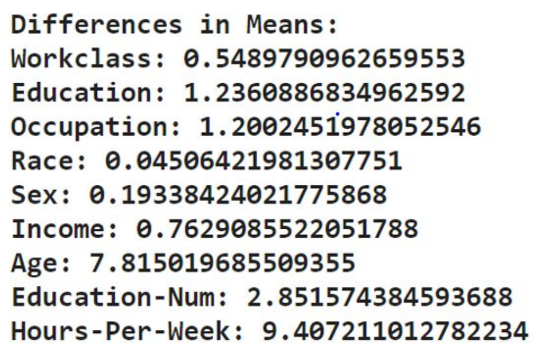


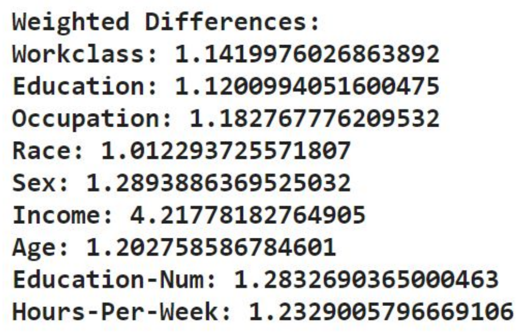
Figure 5 - Plot of All Averages With a Positive Difference From Second Cluster to First.

Following the plot, the next step was to calculate the differences of the Means and determine which was greatest. As *Figure 6* shows, the greatest difference in averages is Hours-Per-Week, with Age in second place. This means that on average, those with higher incomes (roughly $50,000 a year or more) work 9.4 more hours a week and are on average 7.8 years older than those who mostly earn less than $50,000 a year. These numbers are helpful, but do not show the degree of impact these attributes have on income. The next step was to calculate the weight of the differences of each average attribute. We do this by dividing the attribute from the second cluster by the attribute of the first cluster. These weights are shown in *Figure 7* titled 'Weighted Differences'. The weights were then sorted from greatest to least. This is where the most useful results were found and can be seen in *Figure 8*.



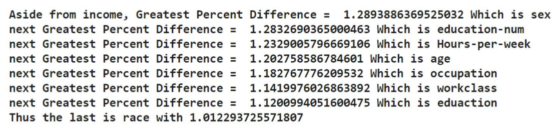
```
Differences in Means:
Workclass: 0.5489790962659553
Education: 1.2360886834962592
Occupation: 1.2002451978052546
Race: 0.04506421981307751
Sex: 0.19338424021775868
Income: 0.7629085522051788
Age: 7.815019685509355
Education-Num: 2.851574384593688
Hours-Per-Week: 9.407211012782234
```

Figure 6 - Difference of Means From Second Cluster to First



```
Weighted Differences:
Workclass: 1.1419976026863892
Education: 1.1200994051600475
Occupation: 1.182767776209532
Race: 1.012293725571807
Sex: 1.2893886369525032
Income: 4.21778182764905
Age: 1.202758586784601
Education-Num: 1.2832690365000463
Hours-Per-Week: 1.2329005796669106
```

Figure 7 - Ratio of Averages From Second Cluster to First (AKA Weighted Differences or Percent Differences)



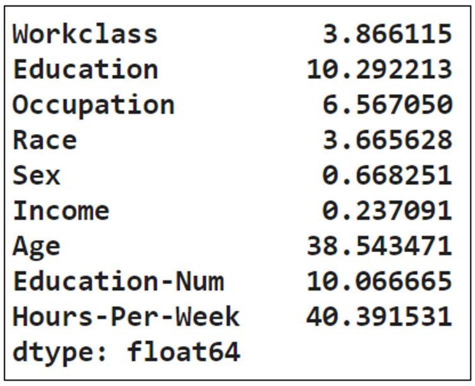
```
Aside from income, Greatest Percent Difference =  1.2893886369525032 Which is sex
next Greatest Percent Difference =  1.2832690365000463 Which is education-num
next Greatest Percent Difference =  1.2329005796669106 Which is Hours-per-week
next Greatest Percent Difference =  1.202758586784601 Which is age
next Greatest Percent Difference =  1.182767776209532 Which is occupation
next Greatest Percent Difference =  1.1419976026863892 Which is workclass
next Greatest Percent Difference =  1.1200994051600475 Which is eduaction
Thus the last is race with 1.012293725571807
```

Figure 8 - Ordering The Ratios From Greatest to Smallest Difference

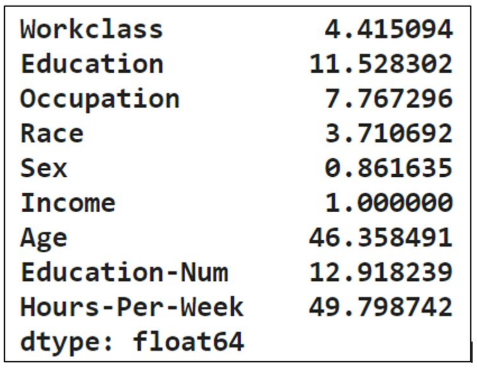
The weighted difference of income is roughly 4.2, the greatest of all the attributes. This is promising information because it shows that K-Means Clustering did well in separating the data in regard to income, thus proving there is some correlation between these attributes and income. Aside from the Income attribute, the greatest weighted difference is Sex with a ratio of 1.29. This could show that, based on our data, the attribute with the greatest impact on income is gender. The next greatest is the Education-Num attribute, which is a numerical representation of one's education level, with a ratio of 1.28. Following that is Hours-Per-Week with a ratio of 1.23, and Age with a ratio of 1.20. Below 1.20 we have, in decreasing order, Occupation, Work-Class, Education, and Race.

We can trust that these results might hold some form of accuracy because, when performing K-Means

Clustering on a dataset containing all attributes except for Income, both clusters have a very clear difference in their Income variables. The first cluster has an average income value of 0.24 and the second cluster has an average income value of 1.00, as seen in *Figure 9.* The average Income difference ratio is the greatest at 1.29.



```
Workclass            3.866115
Education           10.292213
Occupation           6.567050
Race                 3.665628
Sex                  0.668251
Income               0.237091
Age                 38.543471
Education-Num       10.066665
Hours-Per-Week      40.391531
dtype: float64
```



```
Workclass            4.415094
Education           11.528302
Occupation           7.767296
Race                 3.710692
Sex                  0.861635
Income               1.000000
Age                 46.358491
Education-Num       12.918239
Hours-Per-Week      49.798742
dtype: float64
```

Figure 9 - Averages of Attributes From, In Descending Order, First and Second Cluster

These variables were separated all without the use of the attribute in the model, clearly showing a correlation between the predictor attributes and the target attribute. Thus, it is safe to say that K-Means Clustering was a successful model in evaluating the dataset and determining how certain attributes may affect income.

Hierarchical Clustering

After performing K-Means Clustering and having it yield promising results, we decided it was worth it to attempt the same process but with Hierarchical Clustering Instead. Thus we began by clearing the 'Cluster' attribute from the dataset and created our new set with the absence of the target variable, 'Income'. The next thing we did was determine how many clusters we needed by plotting the dendrogram of the data. Using the same logic as we did to explain 2 clusters in K-Means Clustering, we assumed, 2 would also be the optimal number of clusters in our Hierarchical model. However, we plotted the dendrogram to be sure. *Figure 10* shows the dendrogram.

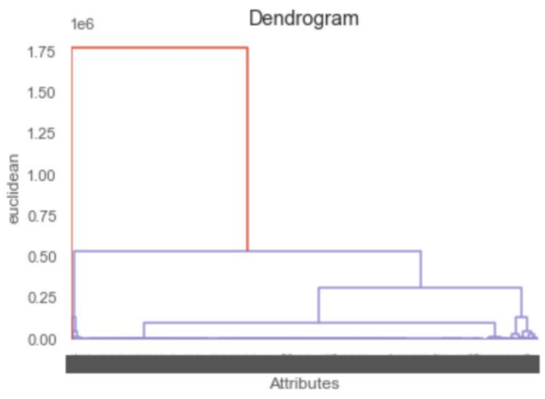
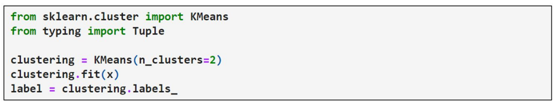
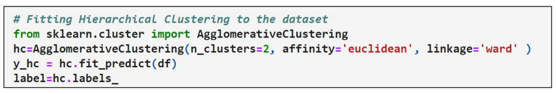


Figure 10 - Dendrogram for Hierarchical Clustering

There are clearly two clusters prominent in the dendrogram, thus we went with two clusters just like we did in K-Means. We fit the model, found the cluster each sample belongs to, and split the data. We calculated the averages, graphed them, and found the differences and ratios. Figure 11 shows how we fit both the K-Means model and the Hierarchical model, thus highlighting that these models are different. *Figure 12* shows our plot of the averages. *Figure 13,* shows the averages, *Figure 14* shows the differences, and lastly *Figure 15* shows the ratios of differences.



```python
from sklearn.cluster import KMeans
from typing import Tuple

clustering = KMeans(n_clusters=2)
clustering.fit(x)
label = clustering.labels_
```



```python
# Fitting Hierarchical Clustering to the dataset
from sklearn.cluster import AgglomerativeClustering
hc=AgglomerativeClustering(n_clusters=2, affinity='euclidean', linkage='ward' )
y_hc = hc.fit_predict(df)
label=hc.labels_
```

Figure 11 - Fitting Models. In Descending Order: K-Means and Hierarchical

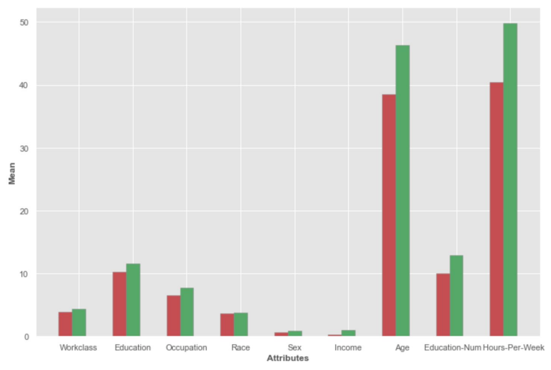
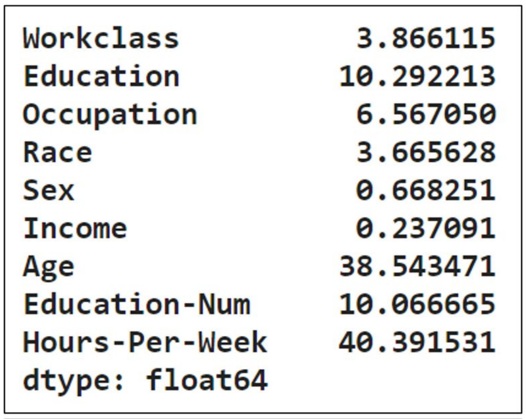
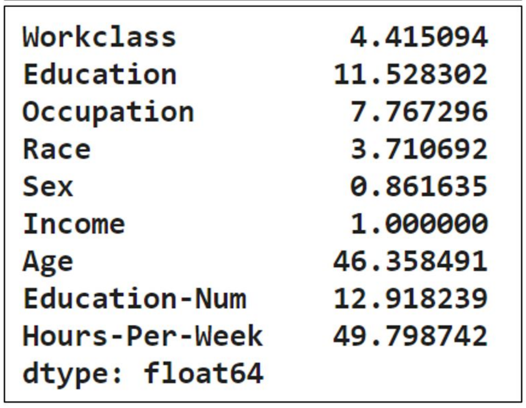


Figure 12 - Averages of Hierarchical Clustering

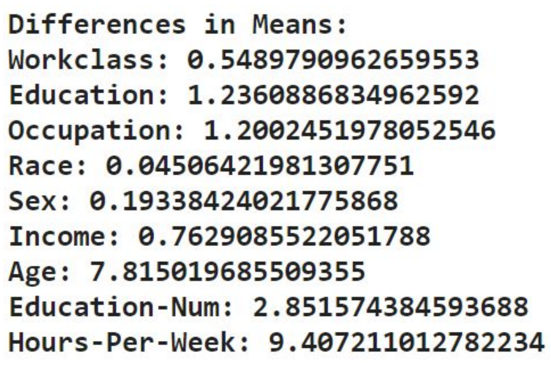


```
Workclass          3.866115
Education         10.292213
Occupation         6.567050
Race               3.665628
Sex                0.668251
Income             0.237091
Age               38.543471
Education-Num     10.066665
Hours-Per-Week    40.391531
dtype: float64
```



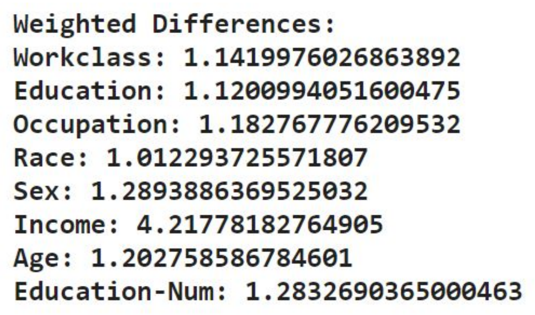
```
Workclass          4.415094
Education         11.528302
Occupation         7.767296
Race               3.710692
Sex                0.861635
Income             1.000000
Age               46.358491
Education-Num     12.918239
Hours-Per-Week    49.798742
dtype: float64
```

Figure 13 - Averages from Hierarchical Clustering. In Descending Order: First and Second Cluster



```
Differences in Means:
Workclass: 0.5489790962659553
Education: 1.2360886834962592
Occupation: 1.2002451978052546
Race: 0.04506421981307751
Sex: 0.19338424021775868
Income: 0.7629085522051788
Age: 7.815019685509355
Education-Num: 2.851574384593688
Hours-Per-Week: 9.407211012782234
```

Figure 14 - Differences of Averages from Hierarchical Clustering



```
Weighted Differences:
Workclass: 1.1419976026863892
Education: 1.1200994051600475
Occupation: 1.182767776209532
Race: 1.012293725571807
Sex: 1.2893886369525032
Income: 4.21778182764905
Age: 1.202758586784601
Education-Num: 1.2832690365000463
```

Figure 15 - Weighted Differences of Averages from Hierarchical Clustering

Comparing these values to those achieved through K-Means Clustering, it is clear to see that both K-Means and Hierarchical Clustering yield the same result. This is most likely due to the fact that only two clusters were used in evaluations, thus creating a limitation on how the samples can be separated. This can also be because clustering in this data yields string results in general, and this is explained in our K-Means conclusion.

Conclusion

Compared to Multivariable Linear Regression, Logistic Regression performed significantly better. The accuracy score we received from the Logistic Regression was at 81% compared to Multivariable Linear Regression, which was at 25%. After performing Logistic Regression we noticed that most of the predicted incomes were under 50,000$ and only few of them were over 50,000$, which is to be expected since the dataset is over twenty years old. The economy in the 90s, compared to the current year, was much different and the income of 50,000$ used to be considered pretty high. Today's equivalent salary would fall just below $90,000. Therefore it's understandable why our data contains mostly incomes that are less than 50K.

Observing the Naive Bayes algorithm it is clear that it performed very well and provided results that were consistent with the initial dataset. After performing an accuracy measurement of the algorithm it was found to have an accuracy of 80%, which lined up nicely with the accuracies of many of the other algorithms. Analyzing the confusion matrix from the predictions made by the Naive Bayes resulted in pretty acceptable results. The variables that required higher values to perform well (true positives and false negatives) achieved values of 95% and 67%. While the values that needed to remain low (false positives and true negatives) were able to remain fairly low with values of 5% and 33%. Looking at the final results of the algorithm 88% of the predicted values correlated to the income level >50k, while 12% correlated to the income level >50k. This further supported the accuracy of the Naive Bayes predictions as a large portion from the original data contained predictions for the income level >50k (approximately ¾ of the original dataset). The Naive Bayes algorithm performed very well attaining a high accuracy and providing consistent results with the dataset.

The results of the Random Forest algorithm although containing a pretty high accuracy score was essentially unusable. Due to how the original data was distributed with a large amount of the predictions being skewed towards the income level of >50k, and the number of features and how split and varied the data was the Random Forest produced results that were difficult to interpret. The

Decision Trees were overly large and almost impossible to read due to the sheer size of the trees. Therefore although the results were correct, because the formatting of the tree was too large and unsuited to the Random Forest or Decision Tree models, the outcome of the algorithms was not useful, and unable to be understood.

Our Clustering algorithms are more examples of models that performed well on our data set. The best way of proving this is to explain that even though the model was fit to a dataset that did not contain the 'Income' attribute, there was a clear difference between both Income attributes in the two separate clusters. Clustering was also successful in helping us find what variables have greater effects on Income than others. More specifically, from our findings, the Sex attribute had the greatest difference between both clusters. Thus, we can conclude that our clustering models found Sex to be one of the attributes, if not the one attribute, with the greatest effect on Income. We also know that this isn't far off from reality, as it is common knowledge that the gender pay gap is a topic of great controversy, with the most popular statistic claiming women earn an average of $0.70 for every $1.00 a man makes. In fact, in our Clustering analysis, we found that the ratio of the second cluster to the first in relation to Sex is 1.29. The inverse of this would represent the percentage of gender difference of the two clusters. The inverse of 1.29 is roughly 0.77, which is close to this popular pay gap statistic. This is just one example that shows the success of our clustering algorithms.

Project Slides:
https://docs.google.com/presentation/d/1tvQDGHooYXJ4pv8hLgpzmk-i60DVUUxmwTm9SHrWdJY/edit#slide=id.p

GitHub Link:
https://github.com/Aaronportal465/DataMiningProject