UVM College Income Prediction Tool - Individual Contribution Report – Jacob Tredwell

Jacob Tredwell

jtredwel@asu.edu

Team 34 (digData)

***Abstract*—This report highlights my efforts in this group project. I helped the team analyze and explore data provided by the United States Census Bureau to determine which Attributes from the 'adult.data' file are important factors for predicting income above or below the critical $50,000 threshold. This report gives an overview of my contributions and what I would provide to the UVW College marketing department to bolster their enrollment process. All my visualizations and classifiers are generated using Python language (pandas, numpy, matplotlib, seaborn, sklearn libraries) and most of my code is listed in the appendix for reference.**

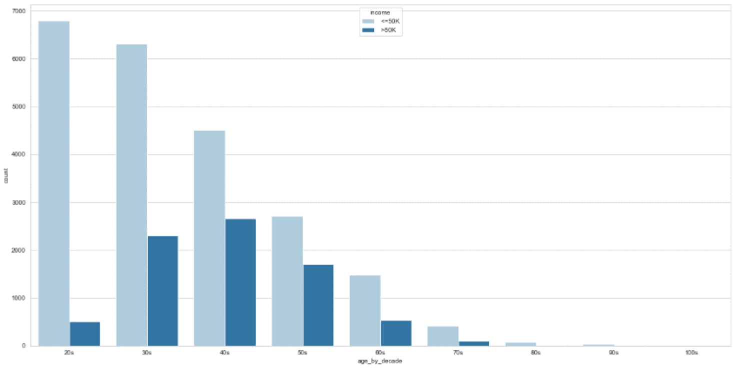
I. My Contribution/Reflection

My initial assignment was to analyze and evaluate whether the 'Age' or 'Workclass' of an individual was relevant to predicting the income for an individual (using $50,000 as the critical threshold). To begin, I produced meaningful visualizations for 'Age' and 'Workclass' using Python and visual variables learned throughout the course.

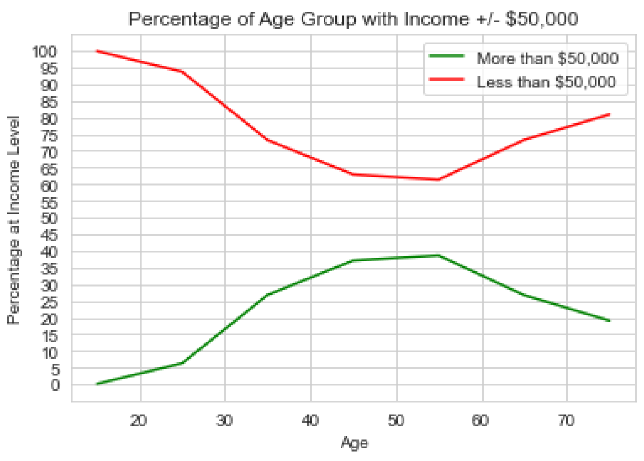
Next, I helped to lead the Zoom meetings and noticed that the 'Capital Gains' feature was not covered by anyone. Based on life-experience, I pointed out that the 'Capital-Gains' feature would likely be a great indicator of income. Therefore, I took it upon myself to research and visualize the Capital Gains data. It turns out that I was right, and Capital Gains ended up being the most salient feature for income prediction.

During initial Zoom meetings, I noticed that many of my team members' visualizations showed the 'Count' of Income-Class for each category. For categories with a small amount of data, the 'bars' in the chart were too small compared to the categories with a larger amount of data. Therefore, I made the suggestion to use the Percentage of each Income-Class that is above or below $50,000. This makes the visualization much easier to interpret and was crucial in our final visualization reporting.

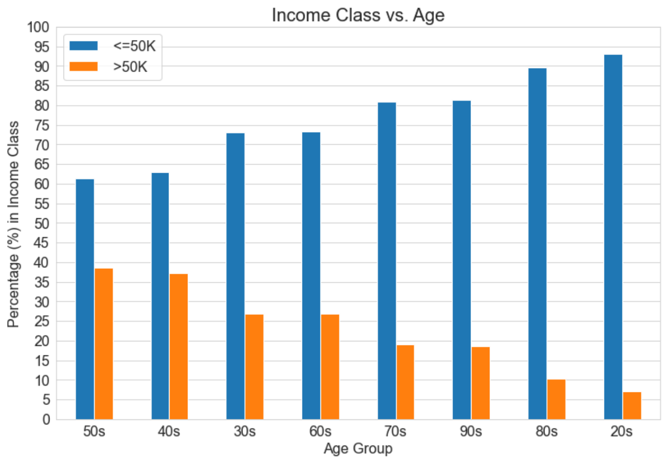
Here are the steps that I took to visualize 'Age' data. I first made a bar chart that showed Count(Income Class) vs. Age Group:



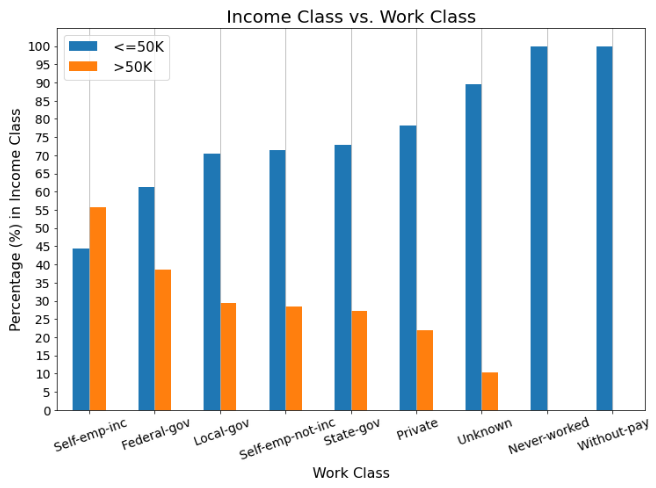
This is not useful, because you cannot see each category (as some counts are small). I switched to percentages and wrote Python code to display a Line Chart (see below).



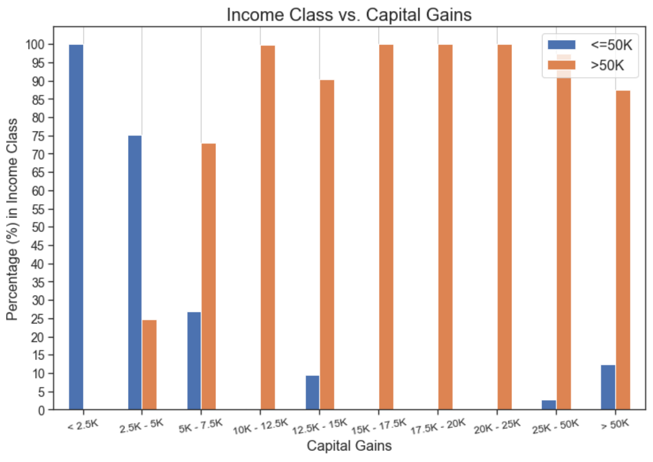
I decided to use a Bar Chart instead (using the seaborn Python library) to plot the Percentage(Income Class) Vs. Age Group sorted by the percentage of individuals making more than $50,000 (with categories in descending order). I believe this is the best visualization technique. It shows that Age is not a terribly effective feature to predict income as every age group has less than a ~38% chance of making more than $50,000 (with many age groups, like those in their 20s and 80s, at a much lower percentage, < 10%).



The next feature I worked on was ‘Workclass’. I used a similar strategy and arrived at a chart for Percentage(Income Class) vs. Workclass. As you can see below, the ‘Workclass’ feature is a better income predictor than ‘Age’:



The final feature I stuided was ‘Capital-Gains’. I used a ‘binning’ strategy to show that Capital Gains is a great indicator of income above or below the critical $50,000 threshold. If an individual’s capital gains is < $5,000 then there is a small chance of making more than $50,000. If Capital Gains are greater than $5,000, there is a very high chance that the individual makes > $50,000 in income. This is the best and most salient attribute for prediction income:



II. Team Overview

All team members collaborated equally on all aspects of the project.

We met on Zoom 3-4 times, usually on Saturday morning, and discussed the relevancy of our features for income prediction and which visualizations are most appropriate for different data types (e.g., categorical vs. continuous-numerical data).

I explained that we want features where specific categories can predict, at a high percentage of accuracy, that an individual’s income is above or below $50,000. We know that most individuals make less than $50,000, so our goal should be to identify which hidden features can best identify the minority of individuals with an income greater than $50,000.

III. What I Learned

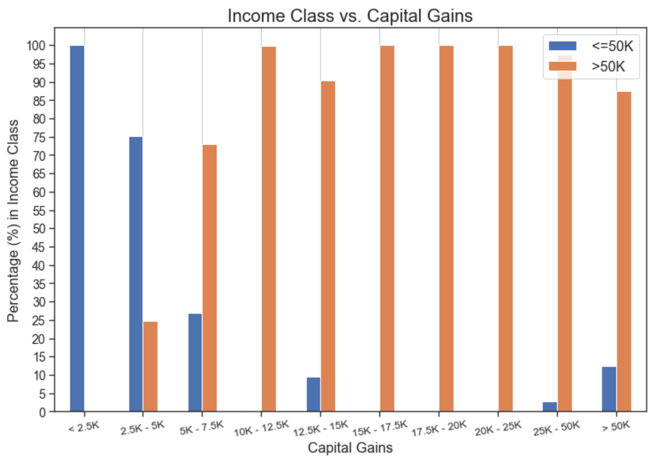
In this project, I expanded my Python language fluency, specifically by using matplotlib, pandas, and seaborn to visualize data. I learned what types of visualizations are most appropriate for different data types.

I also expanded my knowledge of machine learning classification methods and improved my understanding of the ‘sklearn’ library and the

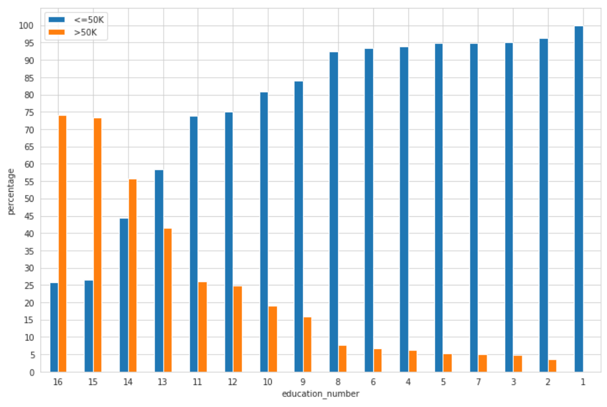
Decision-Tree, K-Nearest Neighbor, and SVM classifiers.

IV. Assessment

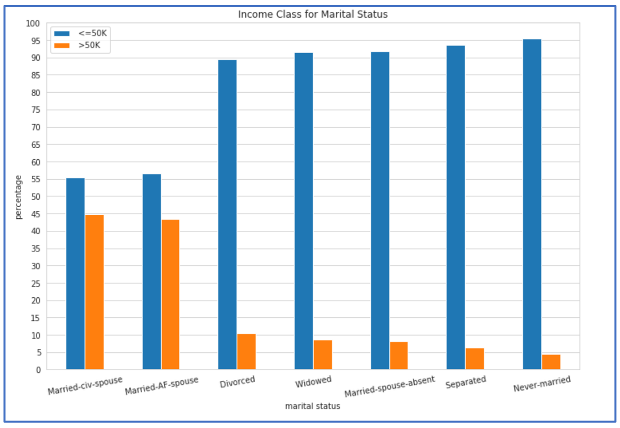
Seeing as **75.9%** of individuals (24,720 out of 32,561 total individuals in the database) **make less than $50,000**, we want features that identify individuals with a higher chance of making **more than $50,000.** Therefore, features that identify the **24.1%** of individuals that make **more than $50,000** will be the most interesting and relevant. To that point, features like **'Capital-Gain'** show a clear trend and identify these individuals.



If an individual has more than $5,000 in Capital Gains, there is a high chance (73-100%) that this individual makes more than $50,000. Conversely, if an individual has less than $5,000 in Capital Gains, there is a high chance (75-100%) that this individual makes less than $50,000.



The **'Education-Num'** feature visualization above shows a similar trend. If an individual has an Education-Num of 14 (Master's degree), 15 (Professional degree), or 16 (Doctorate degree), the individual has a high chance of making more than $50,000. Conversely, if an individual has an Education-Num between 1 and 13, the likelihood that the individual's income is more than $50,000 is quite low.



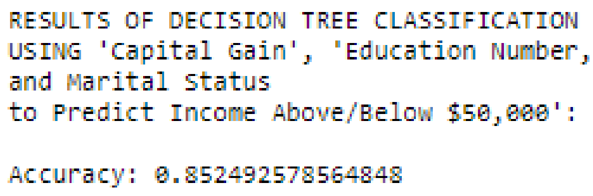
The same trend exists for the **'Marital-Status'** feature (see visualization above). If an individual is **Married** (to a civilian or member of the armed forces), then there is a **much higher chance** that the married individual makes more than $50,000 compared to individuals that are **not married,** as they have an **11% (or less) chance of making more than $50,000**.

V. Future Application

I decided to perform the 'future application' work and include it in this report. I evaluate which methods and features are best for predicting income above or below the critical $50,000 threshold. Then, I use Machine Learning Classification methods along with those salient features to predict Income Class. This application, called **digData Income-Prediction**, is what I will deliver to UVM College. My work is below:

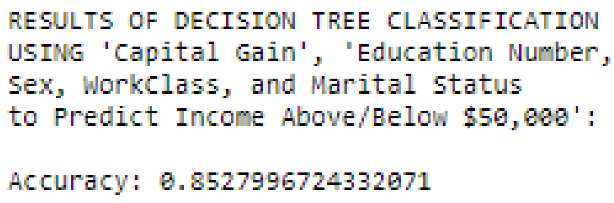
Decision-Tree Classifier:

After studying all the data visualizations for each feature, I determined that **'Capital Gain', 'Marital-Status',** and **'Education-Num'** appear to be the most salient features to use for **predicting** the **target variable** – **'Income-Class**'. As it turns out, these 3 features are indeed important features to predict income. Using the 'DecisionTreeClassifier' model from the 'sklearn.tree' library and the 'train_test_split' function from the 'sklearn.model_selection' library, I was able to predict an whether an individual's income was above/below $50,000 with an **accuracy of 85.249 % (see below):**



```
RESULTS OF DECISION TREE CLASSIFICATION
USING 'Capital Gain', 'Education Number,
and Marital Status
to Predict Income Above/Below $50,000':

Accuracy: 0.852492578564848
```

I can confirm that these 3 features are the most relevant predictors of income, because when I add in two additional features ('Sex' and 'Workclass'), the accuracy only improves by less than 1%. Here are the results when I use a Decision-Tree Classifier using **'Capital Gain', 'Marital-Status', 'Education-Num', 'Sex',** and **'Workclass'** as the predicting features:

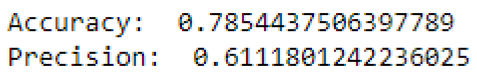


```
RESULTS OF DECISION TREE CLASSIFICATION
USING 'Capital Gain', 'Education Number,
Sex, WorkClass, and Marital Status
to Predict Income Above/Below $50,000':

Accuracy: 0.8527996724332071
```

This is only 0.00030643% better in its prediction, meaning that my hypothesis about 'Capital-Gain', 'Education-Num', and 'Marital-Status' being the most salient features appears to be correct.

SVM Classifier:

Next, I used an SVM classifier using only 'Capital Gain' and 'Education–Num' which gave an Accuracy of 78.544%:

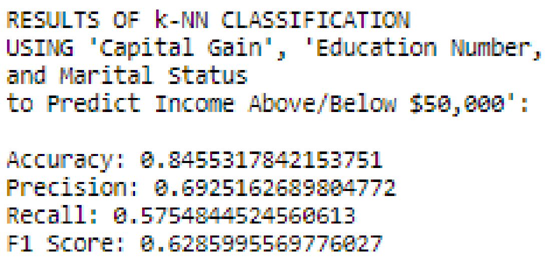


```
Accuracy:  0.7854437506397789
Precision:  0.6111801242236025
```

This further shows that it was a good idea to include 'Marital-Status' as it improved accuracy by about 7%.

K-Nearest Neighbor Classifier:

Similar accuracy results are reached with KNN Classification using the 'Capital-Gain', 'Marital-Status', and 'Education-Num' features to train the model (where k=3):



```
RESULTS OF k-NN CLASSIFICATION
USING 'Capital Gain', 'Education Number,
and Marital Status
to Predict Income Above/Below $50,000':

Accuracy: 0.8455317842153751
Precision: 0.6925162689804772
Recall: 0.5754844524560613
F1 Score: 0.6285995569776027
```

VI. Conclusion

The most salient features from the US Census Bureau Data to predict income are 'Capital-Gain', 'Education-Num', and 'Marital-Status' (with 'Capital-Gain' being the most useful). For each of these 3 features, there is at least one feature-category with a significant percentage of individuals making more than $50,000 (and many categories with a high percentage of individuals making less than $50,000). For the above reason, these are the most relevant features for income prediction. I confirm this fact by training Decision-Tree, SVM, and KNN Classifiers with an income prediction accuracy of over 80% for each. This high level of accuracy will help UVM bolster their enrollment process.

Working with my **digData** team members was a pleasure and it was a useful project that can be used in the real world.

Acknowledgment

Some of the materials and information are extracted from project documentation.

I want to thank Dr. Samira Ghayekhloo for her support and providing lectures and explanations. We also thank Dr. Ross Maciejewski, Dr. K. Selcuk Candan and Dr. Huan Liu for their lectures on Data Visualization and Machine Learning.

REFERENCES

[1] Code for Decision Tree Classifier- Appendix I

[2] SVM Classifier Code - Appendix II

[3] KNN Classifier Code – Appendix III

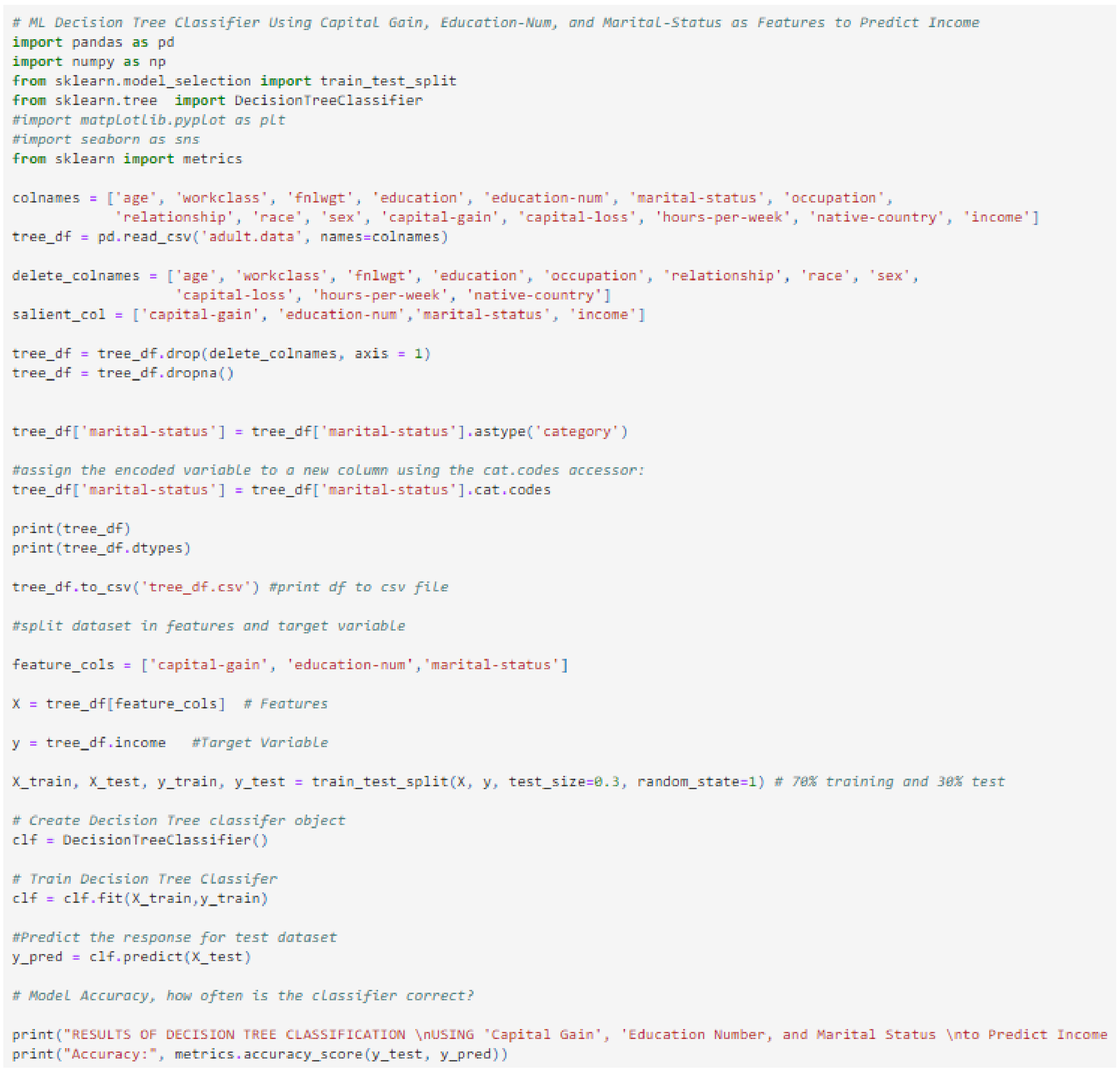
[4] Python code to generate Workclass vs. Income. Appendix IV

[5] Python code to generate Age vs. Income. Appendix V

[6] Python code to generate Capital Gain vs. Income. Appendix VI

[7] Lecture notes.

Appendix I – Code for Decision Tree Classifier

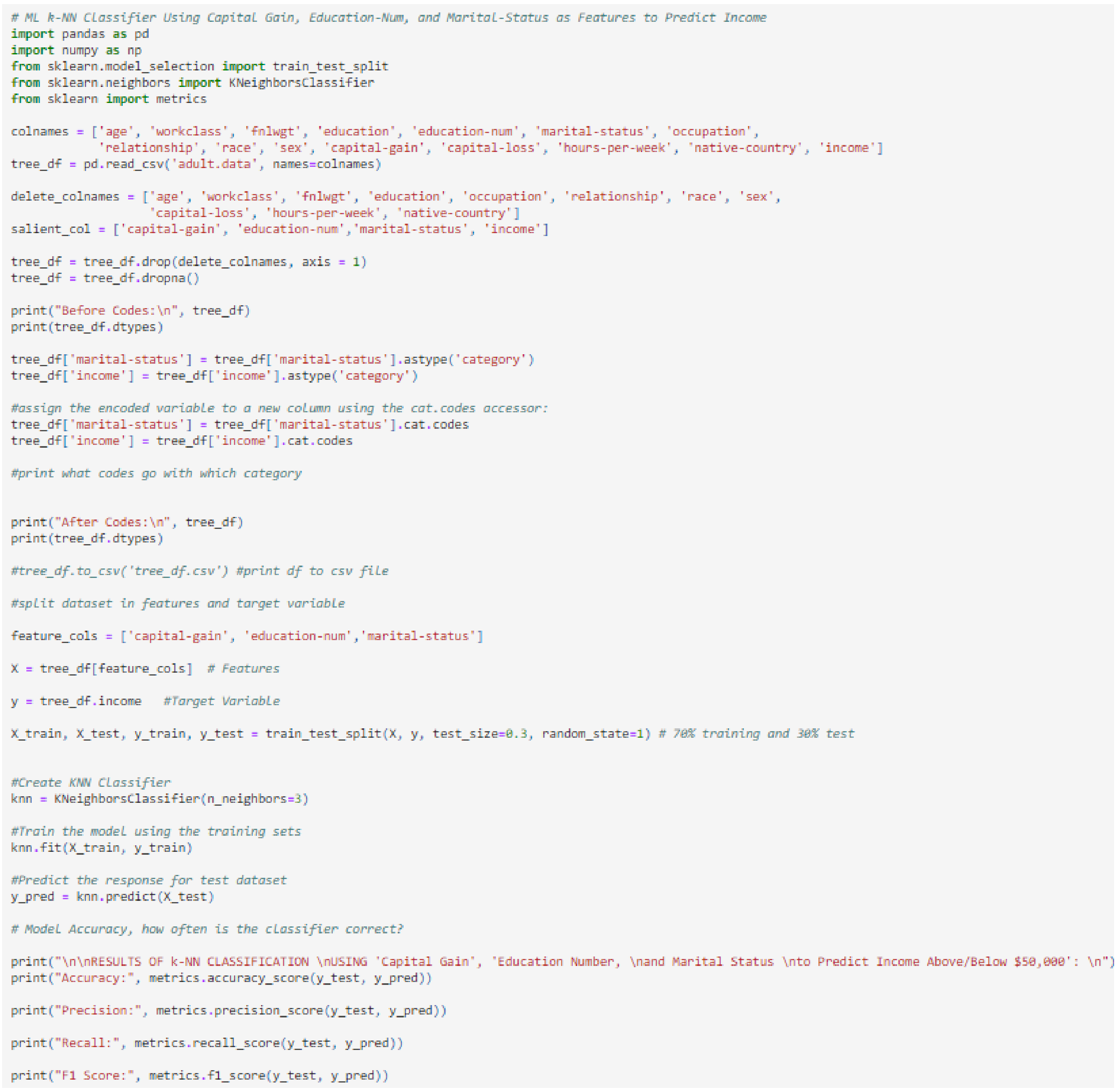


```python
# ML Decision Tree Classifier Using Capital Gain, Education-Num, and Marital-Status as Features to Predict Income
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree  import DecisionTreeClassifier
#import matplotlib.pyplot as plt
#import seaborn as sns
from sklearn import metrics

colnames = ['age', 'workclass', 'fnlwgt', 'education', 'education-num', 'marital-status', 'occupation',
            'relationship', 'race', 'sex', 'capital-gain', 'capital-loss', 'hours-per-week', 'native-country', 'income']
tree_df = pd.read_csv('adult.data', names=colnames)

delete_colnames = ['age', 'workclass', 'fnlwgt', 'education', 'occupation', 'relationship', 'race', 'sex',
                   'capital-loss', 'hours-per-week', 'native-country']
salient_col = ['capital-gain', 'education-num','marital-status', 'income']

tree_df = tree_df.drop(delete_colnames, axis = 1)
tree_df = tree_df.dropna()


tree_df['marital-status'] = tree_df['marital-status'].astype('category')

#assign the encoded variable to a new column using the cat.codes accessor:
tree_df['marital-status'] = tree_df['marital-status'].cat.codes

print(tree_df)
print(tree_df.dtypes)

tree_df.to_csv('tree_df.csv') #print df to csv file

#split dataset in features and target variable

feature_cols = ['capital-gain', 'education-num','marital-status']

X = tree_df[feature_cols]  # Features

y = tree_df.income   #Target Variable

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1) # 70% training and 30% test

# Create Decision Tree classifer object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

# Model Accuracy, how often is the classifier correct?

print("RESULTS OF DECISION TREE CLASSIFICATION \nUSING 'Capital Gain', 'Education Number, and Marital Status \nto Predict Income
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
```

Appendix II – SVM Classifier Code

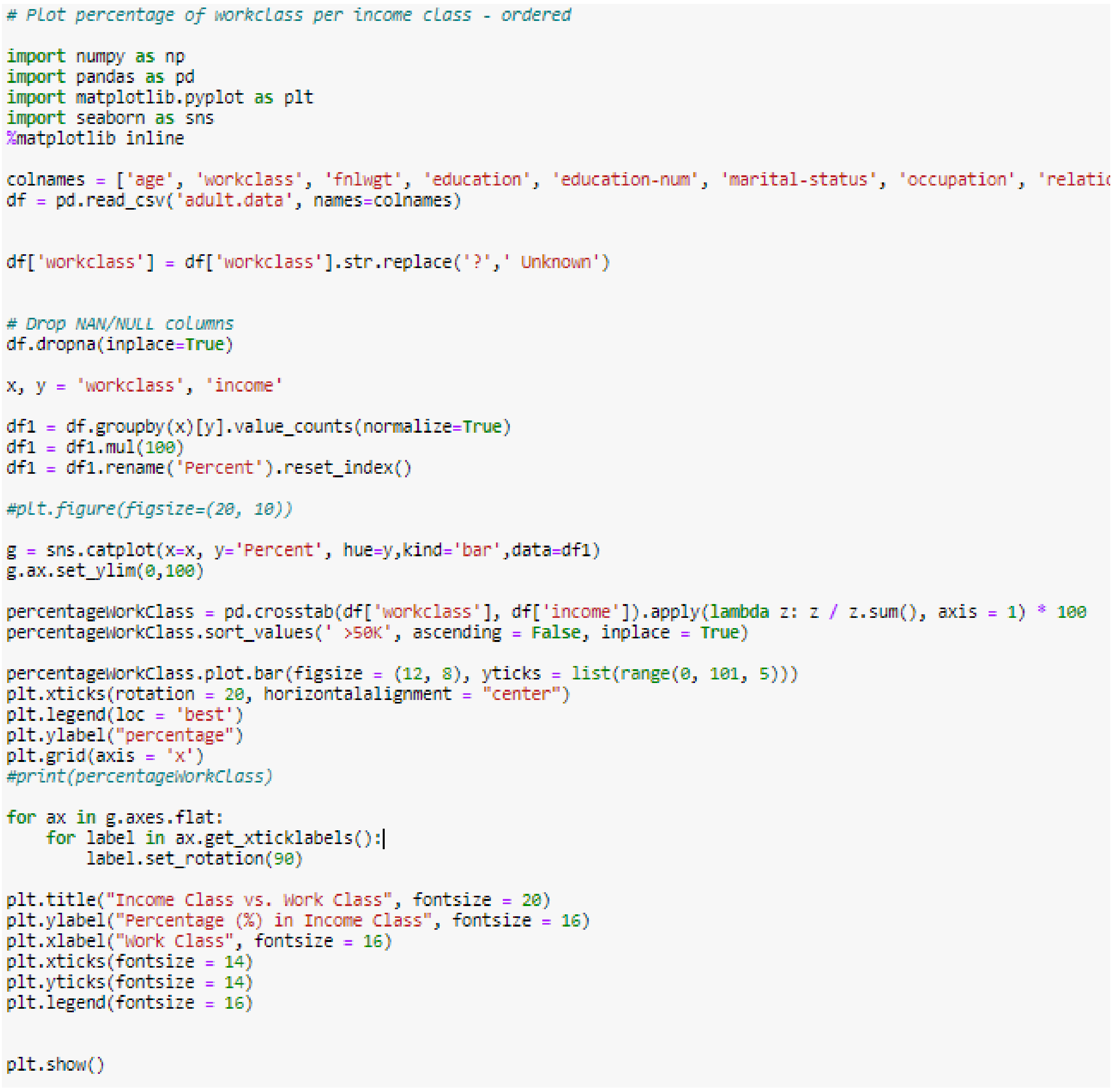
```
# ML SVM Classifier Using Capital Gain and Education-Number as Features to predict income.
import pandas as pd
import numpy as np
from datetime import datetime, timedelta
from sklearn.model_selection import train_test_split
from sklearn.svm  import SVC
import pickle
import matplotlib.pyplot as plt
import seaborn as sns

colnames = ['age', 'workclass', 'fnlwgt', 'education', 'education-num', 'marital-status', 'occupation', 'relationship', 'race', 'sex'
df = pd.read_csv('adult.data', names=colnames)

salient_col = ['capital-gain', 'education-num', 'income']

MLdf = pd.DataFrame()

class_list = []

for i in range(0, len(df)):

    if df['income'][i] == ' <=50K':
        class_list.append(0)

    else:
        class_list.append(1)

#print(class_list)

MLdf['class'] = class_list

MLdf['capital-gain'] = df['capital-gain']

MLdf['education-num'] = df['education-num']

MLdf.to_csv('3_features.csv', index=False)

print(MLdf)



X_train, X_test, y_train, y_test = train_test_split(MLdf.drop(["class"], axis=1), MLdf['class'], test_size = 0.3, random_state = 109)


#new Classifier

clf = SVC(kernel='linear', gamma='auto')



clf.fit(X_train, y_train)  #train the model using training sets

#predict response for test dataset



y_pred = clf.predict(X_test)



from sklearn import metrics

acc = metrics.accuracy_score(y_test, y_pred)

print("Accuracy: ", metrics.accuracy_score(y_test, y_pred))

prec = metrics.precision_score(y_test, y_pred)

print("Precision: ", metrics.precision_score(y_test, y_pred))
```

Appendix III - KNN Classifier Code – Appendix III



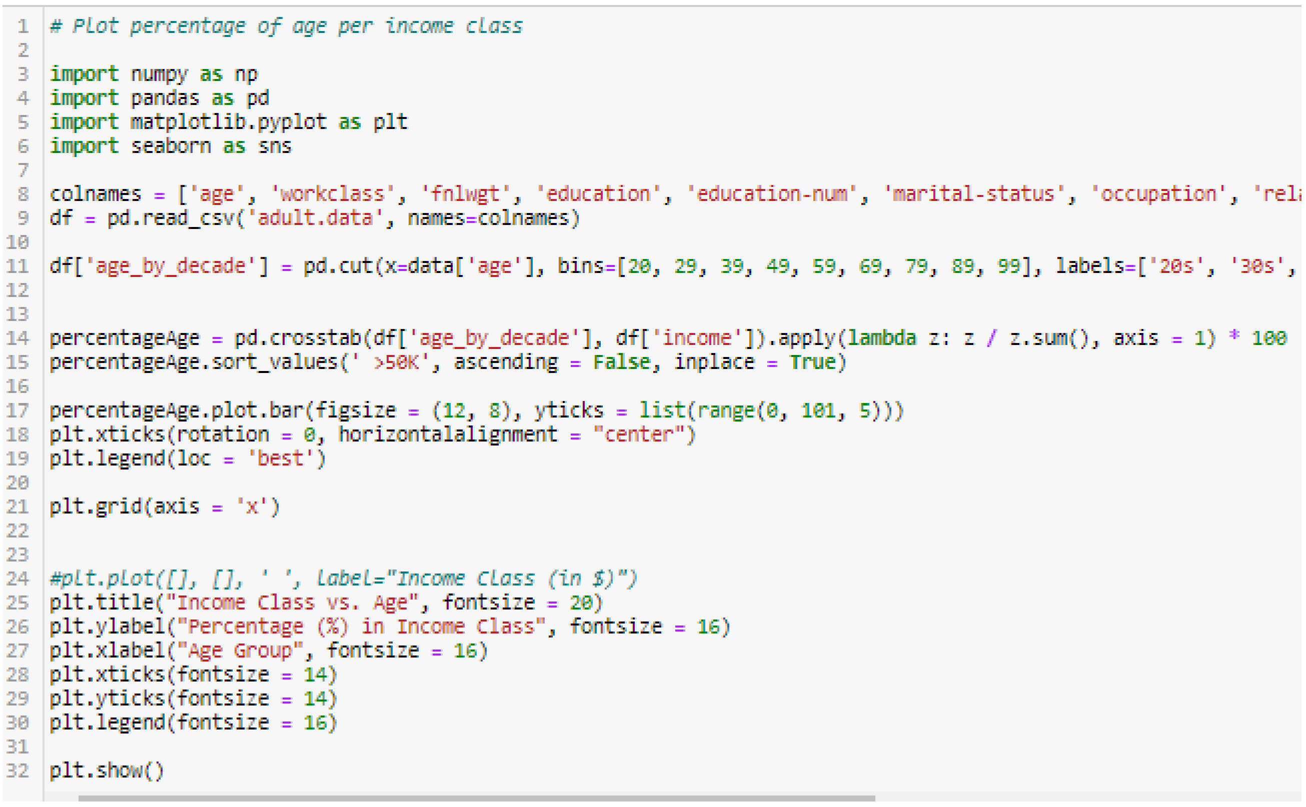
```python
# ML k-NN Classifier Using Capital Gain, Education-Num, and Marital-Status as Features to Predict Income
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics

colnames = ['age', 'workclass', 'fnlwgt', 'education', 'education-num', 'marital-status', 'occupation',
            'relationship', 'race', 'sex', 'capital-gain', 'capital-loss', 'hours-per-week', 'native-country', 'income']
tree_df = pd.read_csv('adult.data', names=colnames)

delete_colnames = ['age', 'workclass', 'fnlwgt', 'education', 'occupation', 'relationship', 'race', 'sex',
                   'capital-loss', 'hours-per-week', 'native-country']
salient_col = ['capital-gain', 'education-num','marital-status', 'income']

tree_df = tree_df.drop(delete_colnames, axis = 1)
tree_df = tree_df.dropna()

print("Before Codes:\n", tree_df)
print(tree_df.dtypes)

tree_df['marital-status'] = tree_df['marital-status'].astype('category')
tree_df['income'] = tree_df['income'].astype('category')

#assign the encoded variable to a new column using the cat.codes accessor:
tree_df['marital-status'] = tree_df['marital-status'].cat.codes
tree_df['income'] = tree_df['income'].cat.codes

#print what codes go with which category


print("After Codes:\n", tree_df)
print(tree_df.dtypes)

#tree_df.to_csv('tree_df.csv') #print df to csv file

#split dataset in features and target variable

feature_cols = ['capital-gain', 'education-num','marital-status']

X = tree_df[feature_cols]  # Features

y = tree_df.income   #Target Variable

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1) # 70% training and 30% test


#Create KNN Classifier
knn = KNeighborsClassifier(n_neighbors=3)

#Train the model using the training sets
knn.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = knn.predict(X_test)

# Model Accuracy, how often is the classifier correct?

print("\n\nRESULTS OF k-NN CLASSIFICATION \nUSING 'Capital Gain", 'Education Number, \nand Marital Status \nto Predict Income Above/Below $50,000': \n")
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))

print("Precision:", metrics.precision_score(y_test, y_pred))

print("Recall:", metrics.recall_score(y_test, y_pred))

print("F1 Score:", metrics.f1_score(y_test, y_pred))
```

Appendix IV – Python code to generate Workclass vs. Income.



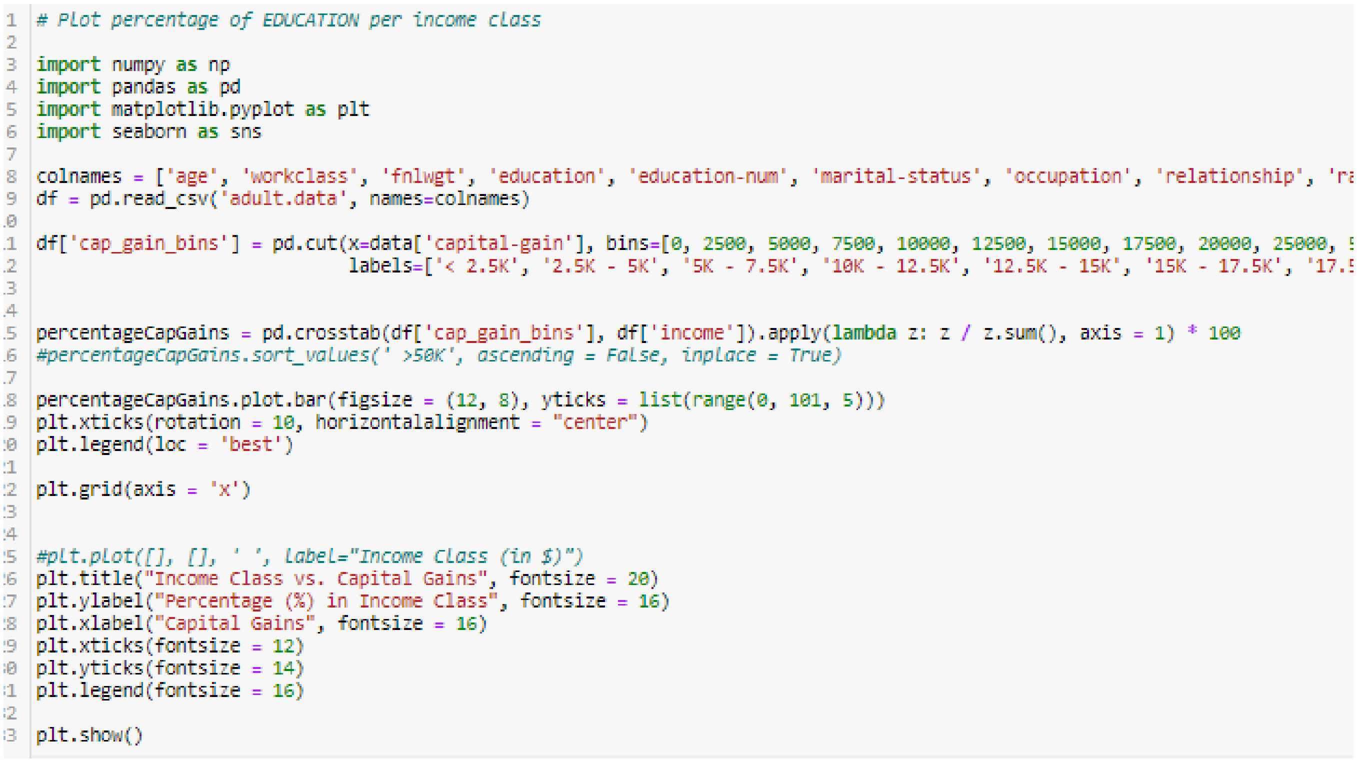
```
# Plot percentage of workclass per income class - ordered

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

colnames = ['age', 'workclass', 'fnlwgt', 'education', 'education-num', 'marital-status', 'occupation', 'relatic
df = pd.read_csv('adult.data', names=colnames)


df['workclass'] = df['workclass'].str.replace('?',' Unknown')


# Drop NAN/NULL columns
df.dropna(inplace=True)

x, y = 'workclass', 'income'

df1 = df.groupby(x)[y].value_counts(normalize=True)
df1 = df1.mul(100)
df1 = df1.rename('Percent').reset_index()

#plt.figure(figsize=(20, 10))

g = sns.catplot(x=x, y='Percent', hue=y,kind='bar',data=df1)
g.ax.set_ylim(0,100)

percentageWorkClass = pd.crosstab(df['workclass'], df['income']).apply(lambda z: z / z.sum(), axis = 1) * 100
percentageWorkClass.sort_values(' >50K', ascending = False, inplace = True)

percentageWorkClass.plot.bar(figsize = (12, 8), yticks = list(range(0, 101, 5)))
plt.xticks(rotation = 20, horizontalalignment = "center")
plt.legend(loc = 'best')
plt.ylabel("percentage")
plt.grid(axis = 'x')
#print(percentageWorkClass)

for ax in g.axes.flat:
    for label in ax.get_xticklabels():
        label.set_rotation(90)

plt.title("Income Class vs. Work Class", fontsize = 20)
plt.ylabel("Percentage (%) in Income Class", fontsize = 16)
plt.xlabel("Work Class", fontsize = 16)
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 14)
plt.legend(fontsize = 16)


plt.show()
```

Appendix V – Python code to generate Age vs. Income.



```
# Plot percentage of age per income class

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

colnames = ['age', 'workclass', 'fnlwgt', 'education', 'education-num', 'marital-status', 'occupation', 'rela
df = pd.read_csv('adult.data', names=colnames)

df['age_by_decade'] = pd.cut(x=data['age'], bins=[20, 29, 39, 49, 59, 69, 79, 89, 99], labels=['20s', '30s',


percentageAge = pd.crosstab(df['age_by_decade'], df['income']).apply(lambda z: z / z.sum(), axis = 1) * 100
percentageAge.sort_values(' >50K', ascending = False, inplace = True)

percentageAge.plot.bar(figsize = (12, 8), yticks = list(range(0, 101, 5)))
plt.xticks(rotation = 0, horizontalalignment = "center")
plt.legend(loc = 'best')

plt.grid(axis = 'x')


#plt.plot([], [], ' ', label="Income Class (in $)")
plt.title("Income Class vs. Age", fontsize = 20)
plt.ylabel("Percentage (%) in Income Class", fontsize = 16)
plt.xlabel("Age Group", fontsize = 16)
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 14)
plt.legend(fontsize = 16)

plt.show()
```

Appendix VI – Python code to generate Capital-Gain vs. Income.



```
# Plot percentage of EDUCATION per income class

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

colnames = ['age', 'workclass', 'fnlwgt', 'education', 'education-num', 'marital-status', 'occupation', 'relationship', 'ra
df = pd.read_csv('adult.data', names=colnames)

df['cap_gain_bins'] = pd.cut(x=data['capital-gain'], bins=[0, 2500, 5000, 7500, 10000, 12500, 15000, 17500, 20000, 25000, 5
                             labels=['< 2.5K', '2.5K - 5K', '5K - 7.5K', '10K - 12.5K', '12.5K - 15K', '15K - 17.5K', '17.5


percentageCapGains = pd.crosstab(df['cap_gain_bins'], df['income']).apply(lambda z: z / z.sum(), axis = 1) * 100
#percentageCapGains.sort_values(' >50K', ascending = False, inplace = True)

percentageCapGains.plot.bar(figsize = (12, 8), yticks = list(range(0, 101, 5)))
plt.xticks(rotation = 10, horizontalalignment = "center")
plt.legend(loc = 'best')

plt.grid(axis = 'x')


#plt.plot([], [], ' ', label="Income Class (in $)")
plt.title("Income Class vs. Capital Gains", fontsize = 20)
plt.ylabel("Percentage (%) in Income Class", fontsize = 16)
plt.xlabel("Capital Gains", fontsize = 16)
plt.xticks(fontsize = 12)
plt.yticks(fontsize = 14)
plt.legend(fontsize = 16)

plt.show()
```