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. All my visualizations and classifiers are generated using Python language (pandas, numpy, matplot.lib, seaborn, sklearn libraries) and most of my code is listed in the appendix for reference.**

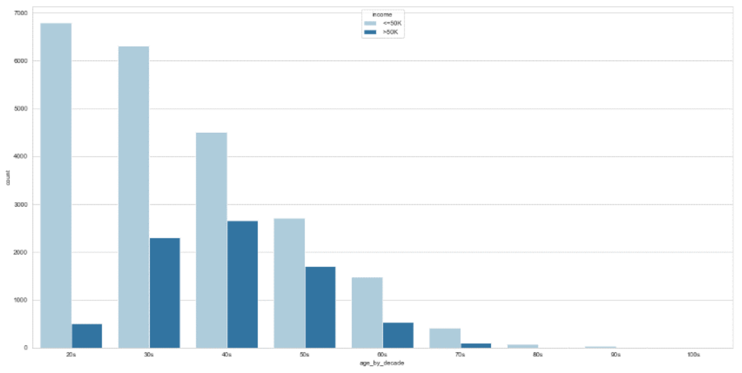
I. My Contribution/Reflection

My initial assignment was to look at ‘Age’ and ‘Workclass’ and produce a meaningful visualization for each to determine whether Age or Workclass were relevant factors to predict income above or below $50,000. I used my data toolkit which consists of many visualization options in Python and visual variables that I learned throughout the course.

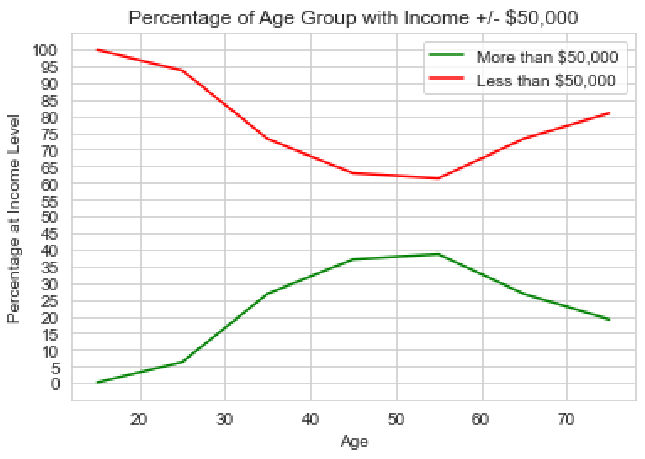
I helped to lead the zoom meetings. I also noticed that the ‘Capital Gains’ feature was not covered by anyone. More importantly, based on life-experience, I pointed out that ‘Capital-Gains’ would likely be a great indicator of income. Therefore, I took it upon myself to research and visualize the Capital-Gain data. It turns out that I was right and Capital gain ended up being the most salient feature for income prediction.

During initial Zoom meetings, I noticed that many of my team member’s visualizations showed the ‘count’ of income class for each category. For categories with a small amount of data, the ‘bars’ in the chart are 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 the Age data. I first made a bar chart that showed Count Income Class vs. Age Group:

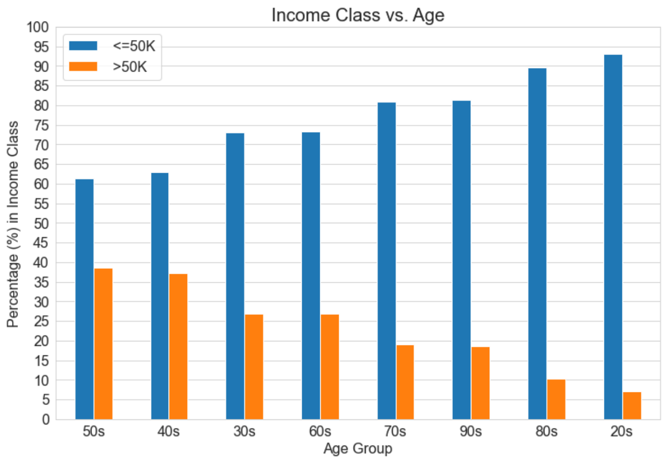


This is not useful, because you cannot see each category (as some counts are small). I switched to percentages and first wrote python code to display a Line Chart (see below).

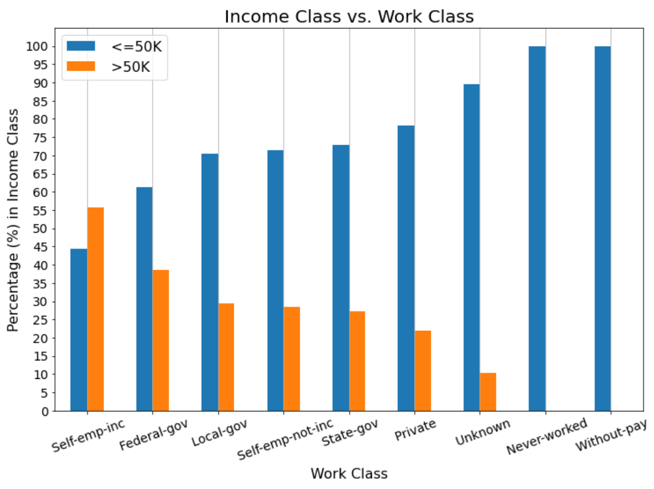


I then wrote code (using the seaborn Python library) to plot the Percentage Income Class Vs. Age Group sorted by Percentage of individuals making more than $50,000 (with categories in descending order). I believe this is the best visualization technique. It also shows that Age is not a terribly good feature to predict income as every age group has less than a

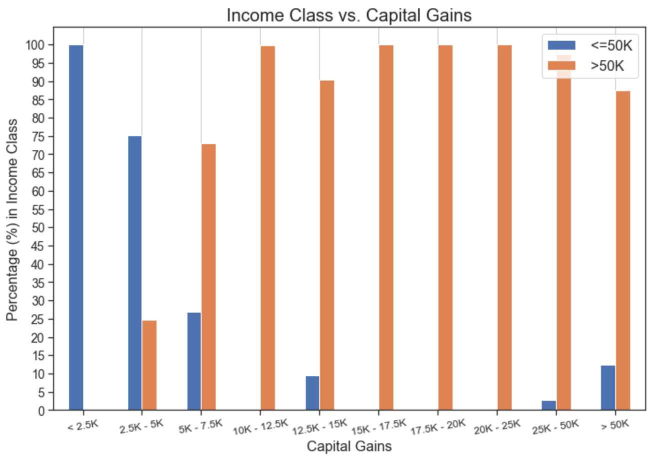
35% chance of making more than $50,000 (with many age groups at a much lower percentage).



The next feature I worked on was Workclass. I used similar strategy as the last visualization for 'Age' and arrived at this graph for % Income Class vs. Workclass. As you can see below, Workclass is a better income predictor than Age.



The last feature is Captial-Gains. I used a binning strategy and this is clearly the most salient feature. 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 > $50k 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 the data type (e.g. categorical vs. continuous-numerical data).

I explained that we want features where specific categories can predict, at a high percentage of accuracy (much greater than 50%), that an individual's income is above or below $50,000.

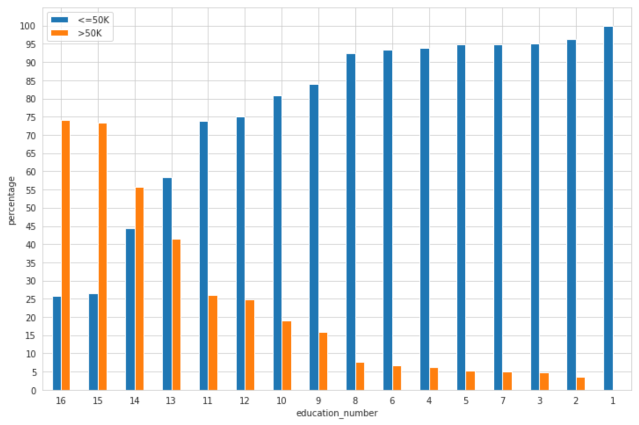
III. What I Learned

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.

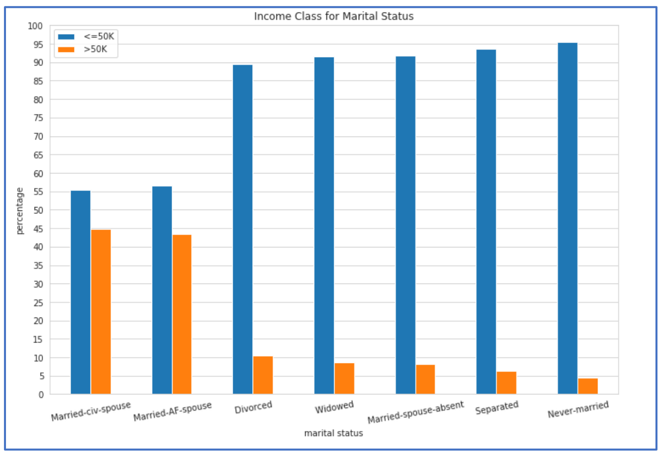
I also expanded my knowledge of machine learning classification methods and improved my understanding of the sklearn library and the Decision Tree and SVM classifiers.

IV. Assessment

From the feature visualizations, I look for attributes where, in each category, there are homogeneous groups where almost everyone in the group belongs to the same income class – this is what will be useful in our income prediction tool. We want features that have categories of individuals with a high chance of making **more than $50,000.** Most people make less than $50,000, so the greater than $50,000 categories are most interesting. Specifically, Features like **'Capital Gain'** show a clear trend. If an individual has more than $5,000 in Capital Gains, there is a very high chance that individual makes more than $50,000 in income. Conversely, if and individual has Capital Gains of less than $5000, there is a high chance that the individual makes less than $50,000.



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



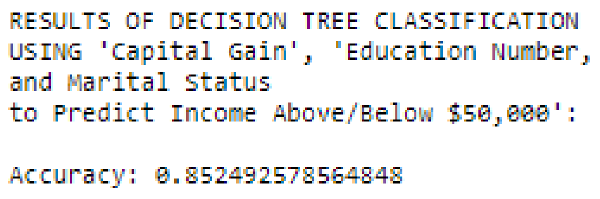
The same trend exists for the **'Marital-Status'** feature (see visualization above). If an individual is **Married** (with a 'civilian' spouse or married with a spouse that is a member of the 'armed-forces'), then there is a **much higher chance** that the married individual makes more than $50,000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

V. Future Application

I decided that I wanted to perform the future application to include in this report. The future application is to use Machine Learning Classification methods and various salient features to predict income class. Then, I evaluate which methods and features are best for predicting income above or below the critical $50,000 threshold. This work what I will give to UVM College. My work is below:

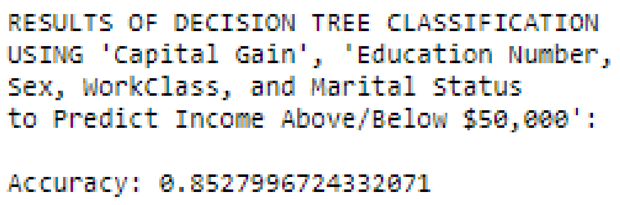
Decision Tree Classifier:

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 DecisionTreeClassifier from sklearn.tree and train_test_split from sklearn.model_selection, 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 a fraction of 1 percent. Here are the results when I use a Decision-Tree Classifier using **'Capital Gain', 'Marital-Status', 'Sex', 'Workclass',** and **'Education-Num'** as the predicting features:

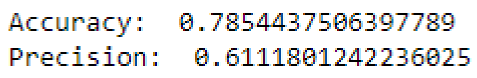


```
RESULTS OF DECISION TREE CLASSIFICATION
USING 'Capital Gain', 'Education Number,
Sex, WorkClass, and Marital Status
to Predict Income Above/Below $50,000':

Accuracy: 0.8527996724332071
```

This is only 0.00030643 % better in its prediction. That means that my hypothesis about ‘Capital-Gain’, ‘Marital Status’, and ‘Marital-Status’ being the most salient features appears to be correct.

SVM Classifier:

I also 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%.

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 salient). For these 3 features, there are homogeneous categories with low-entropy and at least one category with a high percentage making > $50,000. This makes them relevant features for income prediction. I confirm this by training a Decision Tree Classifier with an income prediction accuracy of over 85%.

Working with my digData team members was a pleasure and it was an interesting and 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

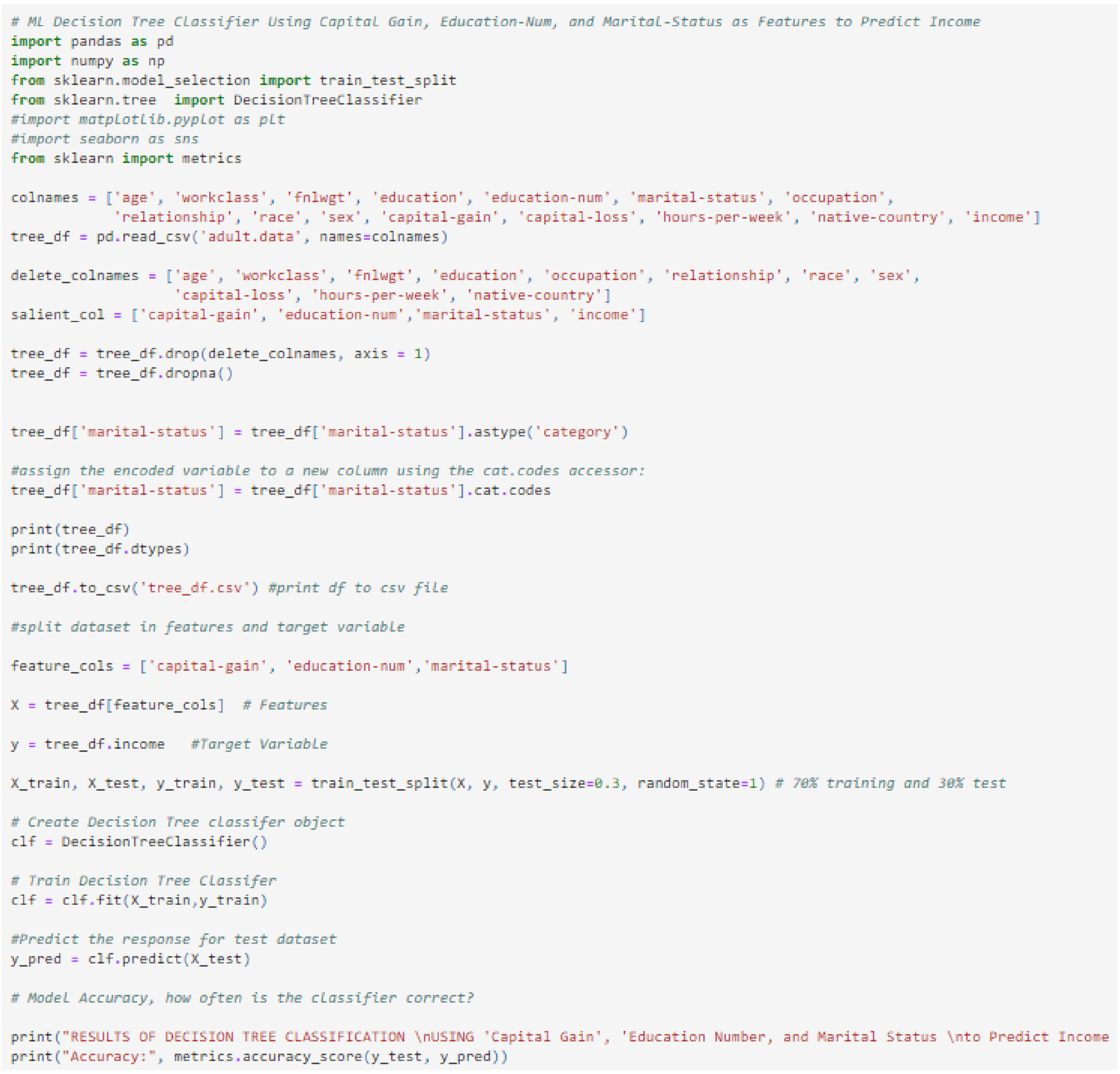
[3] Python code to generate Workclass vs. Income. Appendix III

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

[5] Python code to generate Capital Gain vs. Income. Appendix IV

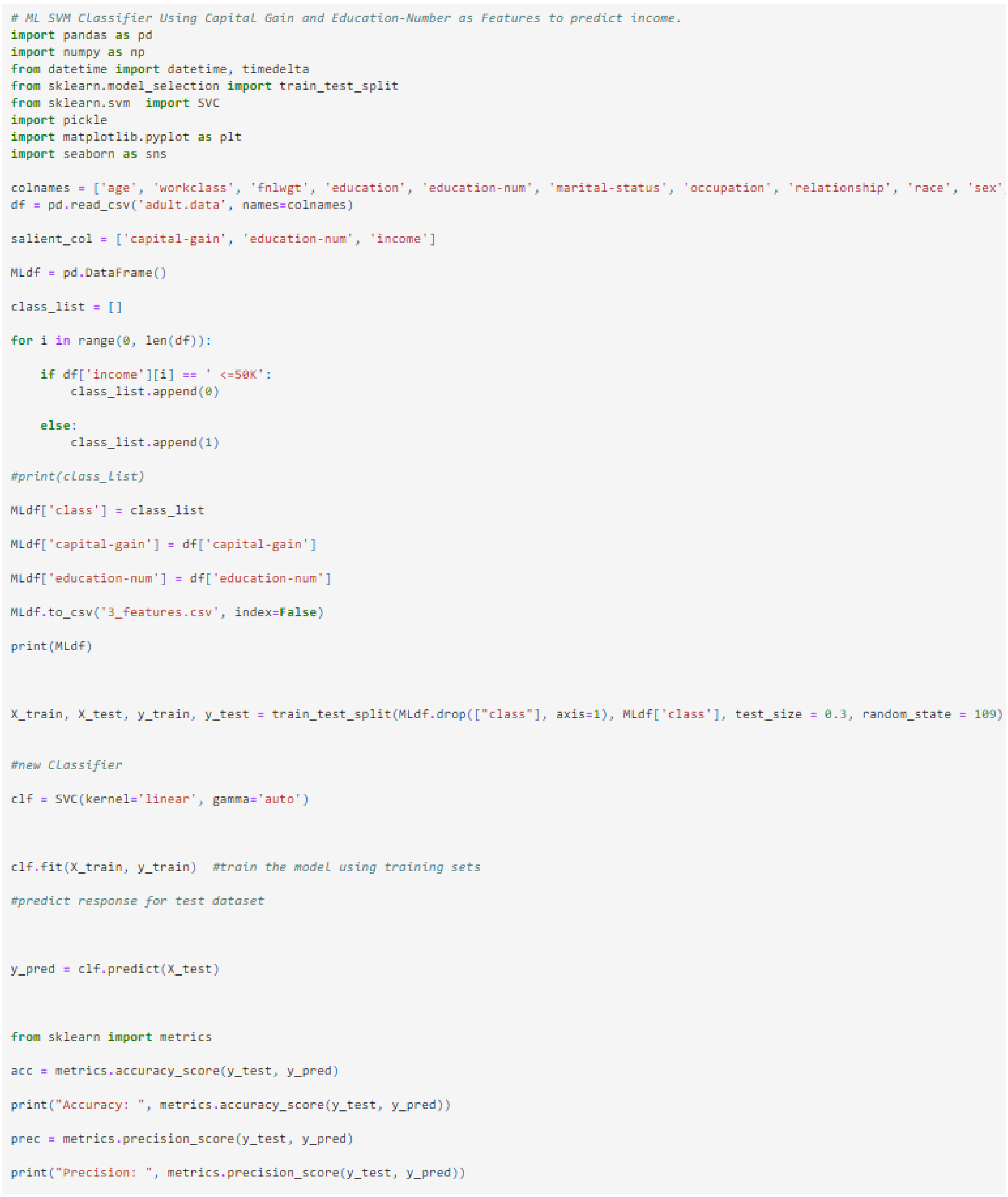
[6] Lecture notes.

Appendix I – Code for Decision Tree Classifier



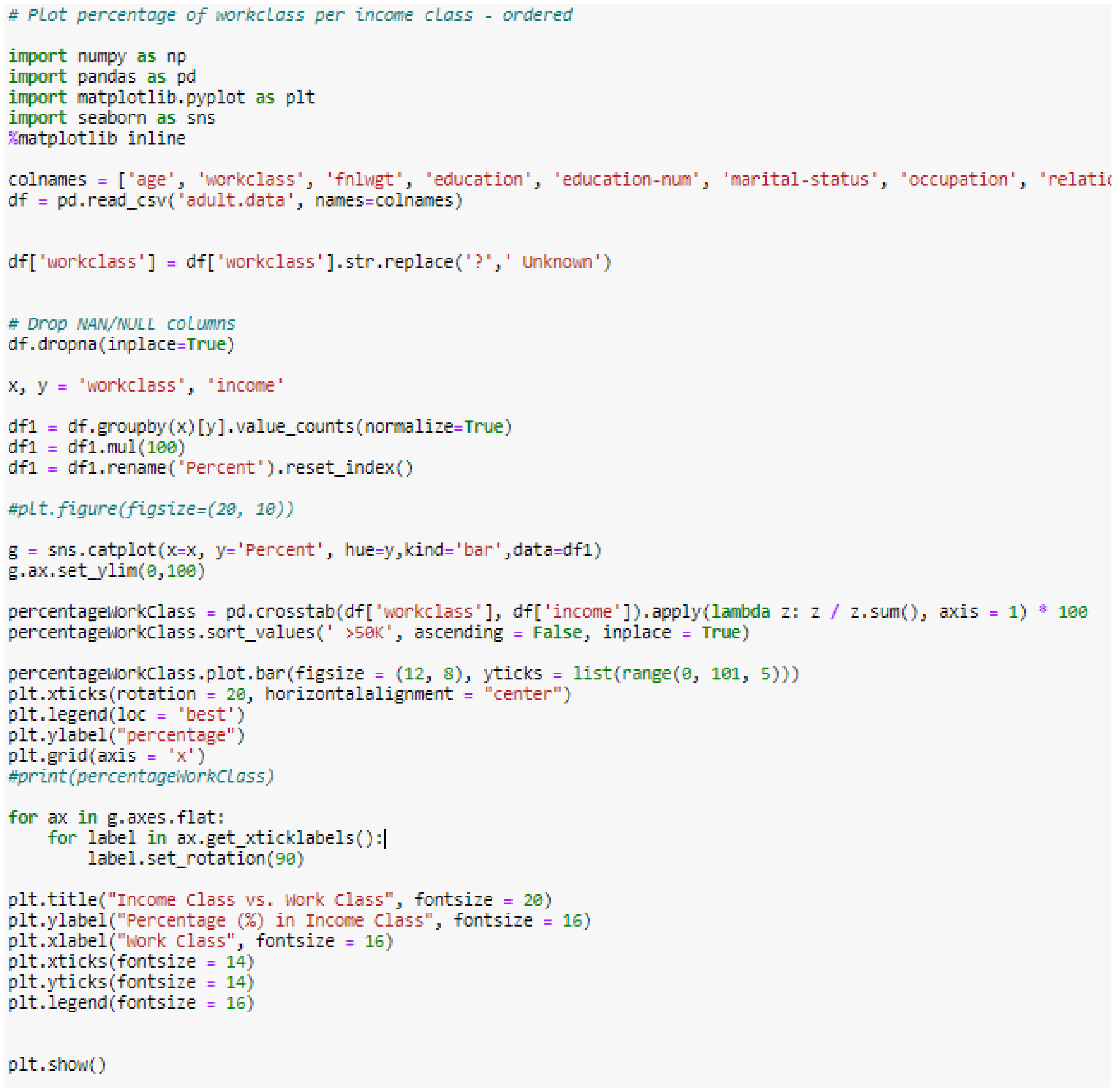
```python
# ML Decision Tree Classifier Using Capital Gain, Education-Num, and Marital-Status as Features to Predict Income
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree  import DecisionTreeClassifier
#import matplotlib.pyplot as plt
#import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



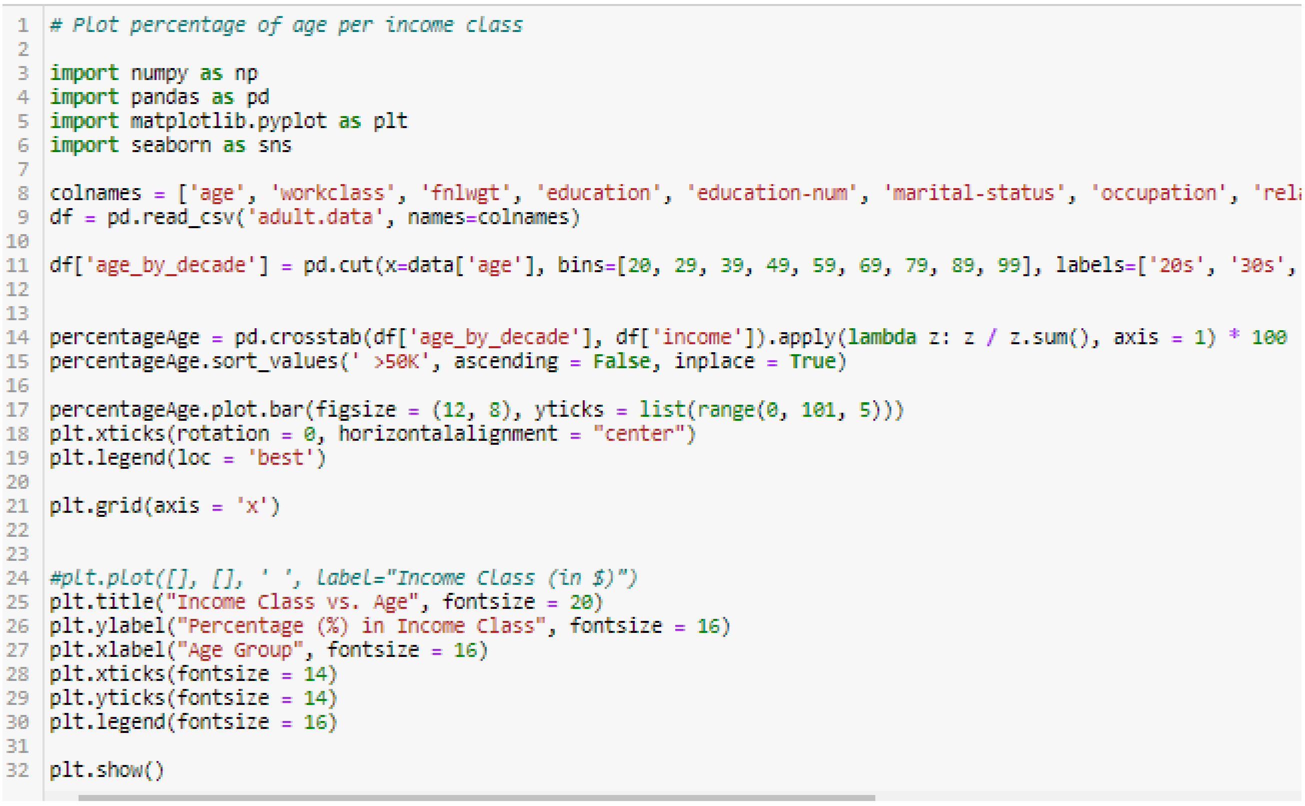
```python
# ML SVM Classifier Using Capital Gain and Education-Number as Features to predict income.
import pandas as pd
import numpy as np
from datetime import datetime, timedelta
from sklearn.model_selection import train_test_split
from sklearn.svm  import SVC
import pickle
import matplotlib.pyplot 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 – Python code to generate Workclass vs. Income.



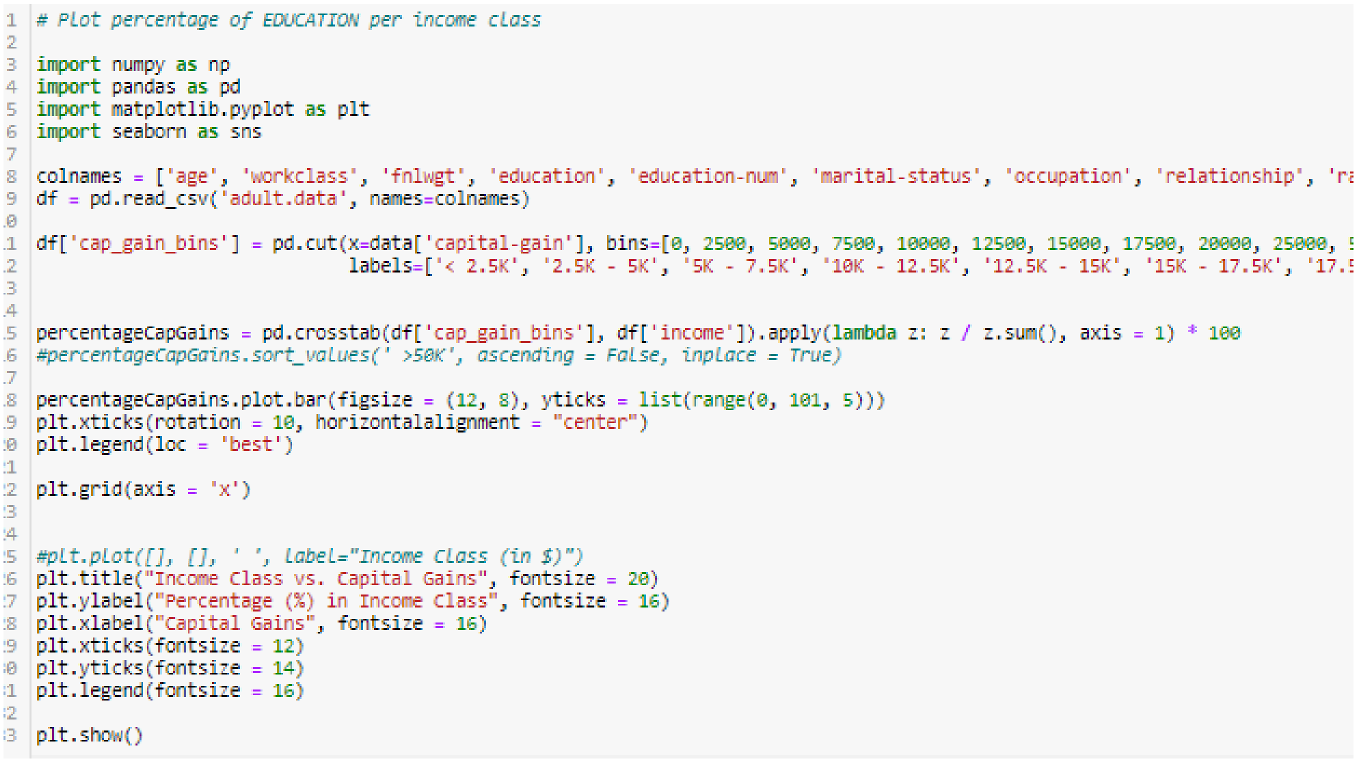
```
# Plot percentage of workclass per income class - ordered

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', 'relati
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 IV – 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 V – 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()
```