Module 1 Exercises for Python

Filipp Krasovsky, 3-8-21

In [2]:
```
#import basics
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Questions for Chapter 4:#21, 22, 23, 24, & 25

Dataset to use: bank_marketing_training

In [3]:
```
bank = pd.read_csv("bank_marketing_training")
```

In [4]:
```
#sanity check
bank.head()
```

Out[4]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	days_since_previous	pre
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	
2	41	blue-collar	married	unknown	unknown	no	no	telephone	may	mon	...	1	999	
3	25	services	single	high.school	no	yes	no	telephone	may	mon	...	1	999	
4	29	blue-collar	single	high.school	no	no	yes	telephone	may	mon	...	1	999	

5 rows × 21 columns

Question 21:

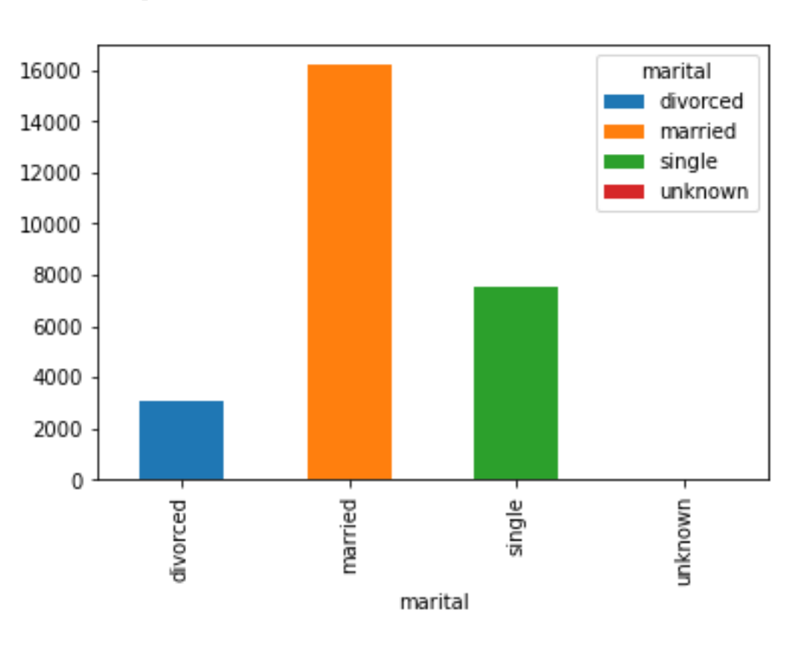
a. Bar graph of marital.

b. Bar graph of marital, with overlay of response.

c. Normalized bar graph of marital, with overlay of response.

In [5]:
```
#bar graph of marital
bargraph_1 = pd.crosstab(bank['marital'],bank['marital'])
bargraph_1.plot(kind='bar',stacked=True)
```

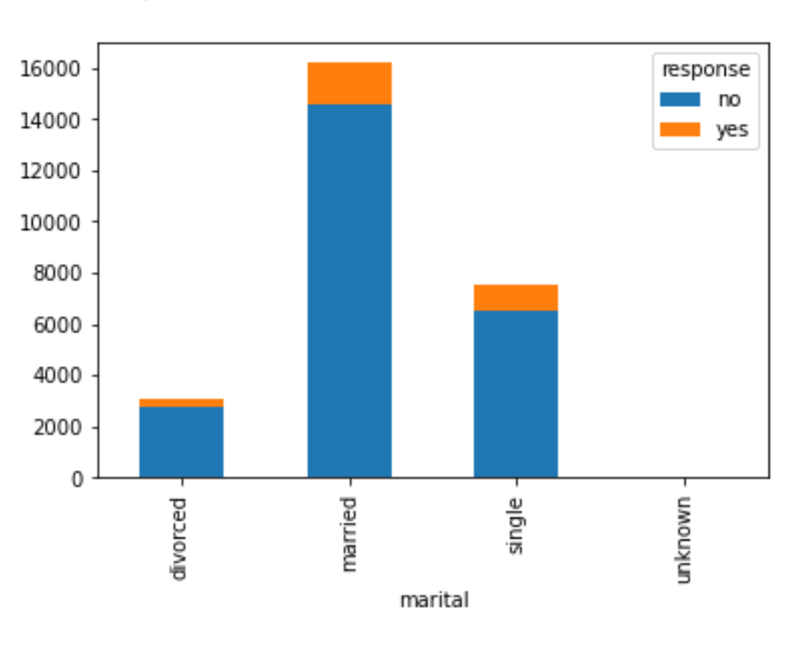
Out[5]:
```
<AxesSubplot:xlabel='marital'>
```



Strength: Shows us raw distribution of variable for further analysis Weakness: Does not show relationship to target variable.

In [6]:
```
#overlayed bar graph
bargraph_2 = pd.crosstab(bank['marital'],bank['response'])
bargraph_2.plot(kind='bar',stacked=True)
```

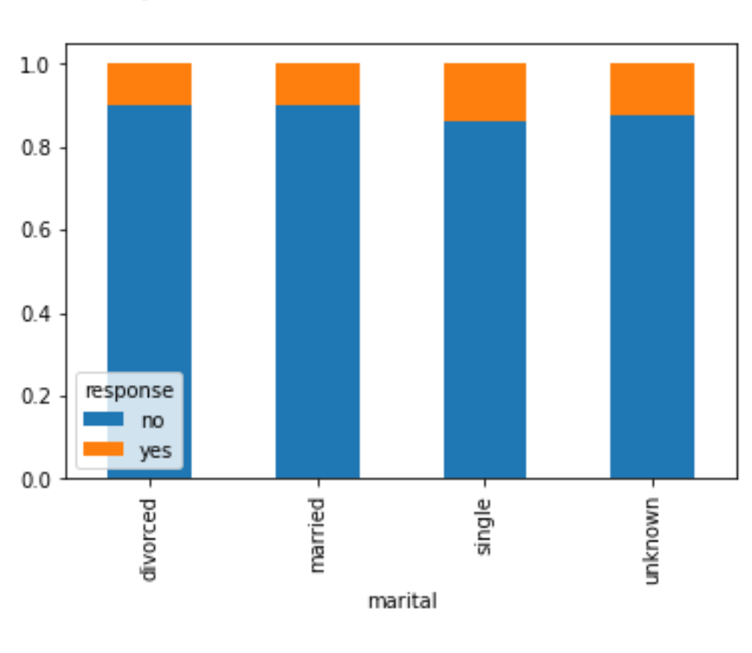
Out[6]:
```
<AxesSubplot:xlabel='marital'>
```



Strength: Shows relationship with target variable Weakness: Lack of normalization obscures relationship

In [7]:
```
#normalized overlayed bar graph
crosstab_01 = pd.crosstab(bank['marital'],bank['response'])
crosstab_norm = crosstab_01.div(crosstab_01.sum(1),axis=0) #sum(1) = sum of the rows of the table, axis
 = 0 -> divide each row by this value.
crosstab_norm.plot(kind='bar',stacked=True)
```

Out[7]:
```
<AxesSubplot:xlabel='marital'>
```



Strength: Normalization makes relationship between response and predictor variables clearer. Weakness: Does not show us distribution of predictor variable.

Question 22:

Using the graph from Exercise 21c, describe the relationship between marital and response.

Since the 'unknown' marital status does not provide much insight on face value, we are left comparing single, married, and divorced individuals. In these instances, Single individuals have a significantly higher rate of responding "yes" than married or divorced individuals. That being said, individuals who are not identifiably married or divorced also had a higher response rate of "yes".

Question 23: Do the following with the variables marital and response.

a. Build a contingency table, being careful to have the correct variables representing the rows and columns. Report the counts and the column percentages.

b. Describe what the contingency table is telling you.

In [8]:
```
#contingency table
crosstab_02 = pd.crosstab(bank['response'], bank['marital'])
#getting percentages:
crosstab_02_percent = round(crosstab_02.div(crosstab_02.sum(0), axis = 1)*100, 1)

print(crosstab_02)
print(crosstab_02_percent)
```

```
marital   divorced  married  single  unknown
response
no            2743    14579    6514       50
yes            312     1608    1061        7
marital   divorced  married  single  unknown
response
no            89.8     90.1    86.0     87.7
yes           10.2      9.9    14.0     12.3
```

The contingency table shows us the same story, largely, as the graph - Single people and those with no clear marital status were more likely to respond "yes" than their married and divorced counterparts by 2-5%, depending on the comparrison.

Question 24: Repeat the previous exercise, this time reporting the row percentages. Explain the difference between the interpretation of this table and the previous contingency table.

In [9]:
```
#contingency table
crosstab_03 = pd.crosstab(bank['response'], bank['marital'])
#getting percentages:
crosstab_03_percent = round(crosstab_03.div(crosstab_02.sum(1), axis = 0)*100, 1)

print(crosstab_03)
print(crosstab_03_percent)
```

```
marital   divorced  married  single  unknown
response
no            2743    14579    6514       50
yes            312     1608    1061        7
marital   divorced  married  single  unknown
response
no            11.5     61.0    27.3      0.2
yes           10.4     53.8    35.5      0.2
```

The difference in interpretation is that here we're examining the demographic composition of those who said "yes" and "no", versus the analysis of the previous table, which was concerned with identifying the portion of each marital group that responded a certain way. Here, we can report that the largest portion of individuals who responded "no" and "yes" were both married people, while in the previous report we identified that single people are more likely to respond "yes".

Question 25. Produce the following graphs. What is the strength of each graph? Weakness?

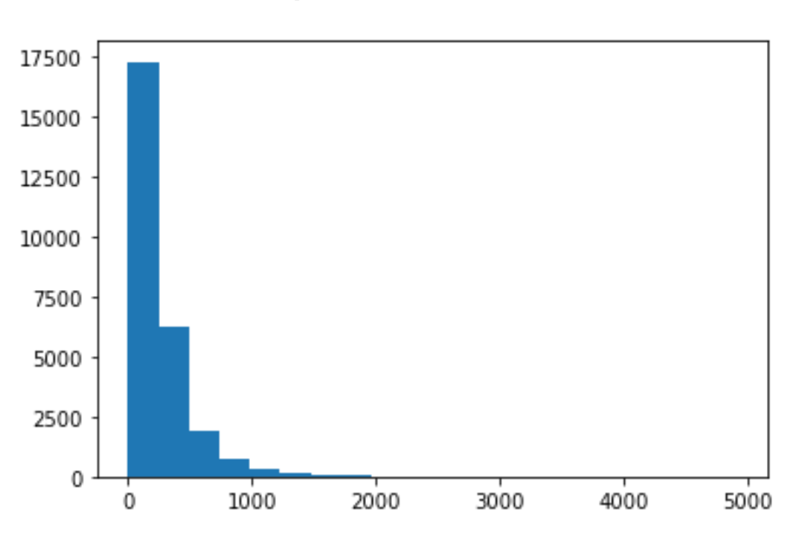
a. Histogram of duration.

b. Histogram of duration, with overlay of response.

c. Normalized histogram of duration, with overlay of response.

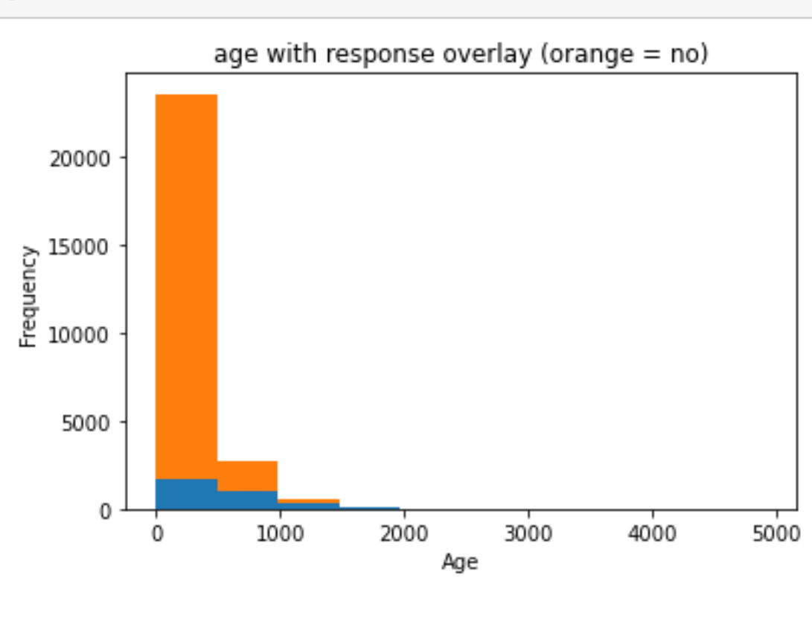
In [10]:
```
#a histogram of duration
plt.hist(bank['duration'],bins=20)
```

Out[10]:
```
(array([1.7308e+04, 6.2620e+03, 1.9260e+03, 7.4100e+02, 3.4100e+02,
        1.5400e+02, 6.4000e+01, 3.2000e+01, 2.1000e+01, 8.0000e+00,
        5.0000e+00, 1.0000e+00, 1.0000e+00, 5.0000e+00, 2.0000e+00,
        1.0000e+00, 0.0000e+00, 1.0000e+00, 0.0000e+00, 1.0000e+00]),
 array([   0. ,  245.9,  491.8,  737.7,  983.6, 1229.5, 1475.4, 1721.3,
        1967.2, 2213.1, 2459. , 2704.9, 2950.8, 3196.7, 3442.6, 3688.5,
        3934.4, 4180.3, 4426.2, 4672.1, 4918. ]),
 <BarContainer object of 20 artists>)
```



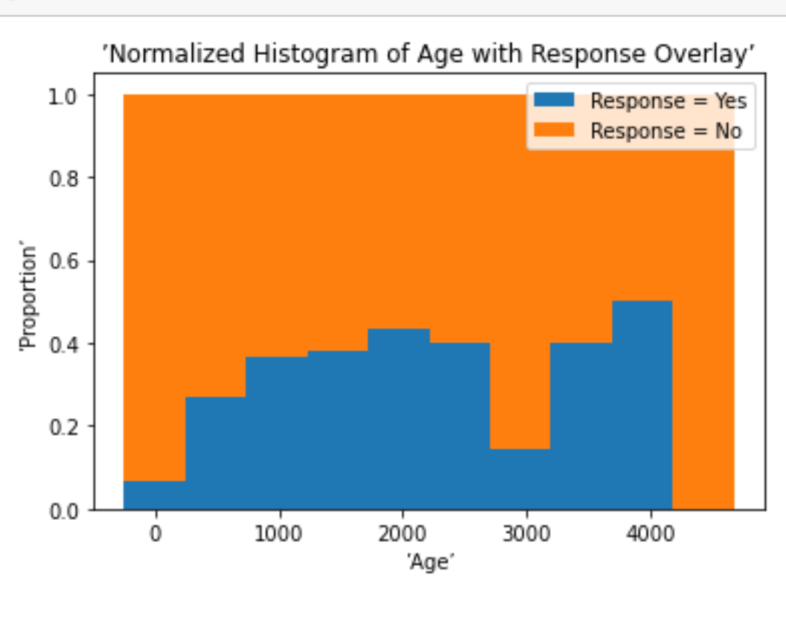
Strength: Shows us distribution of raw data weakness: does not show relationship to target variable

In [17]:
```
#b histogram of duration with response overlay
#separate our response value by all possible variables
bank_train=bank
bt_age_y = bank_train[bank_train.response == "yes"]['duration']
bt_age_n = bank_train[bank_train.response == "no"]['duration']

plt.hist([bt_age_y, bt_age_n], bins = 10, stacked = True)
plt.title('age with response overlay (orange = no)')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



Strength: Shows relationship with target variable Weakness: Does not show proportion of each bin belonging to subgroup of the response variable.

In [22]:
```
#normalized overlay
#normalization process - save the height of the histogram to n and the boundaries of each bin in the hi
stogram to bins.
#(n, bins, patches) = plt.hist([bt_age_y, bt_age_n], bins =10, stacked = True)

#normalize
n_table = np.column_stack((n[0], n[1]))
n_norm = n_table / n_table.sum(axis=1)[:, None]
ourbins = np.column_stack((bins[0:10], bins[1:11]))

p1 = plt.bar(x = ourbins[:,0], height = n_norm[:,0],width = ourbins[:, 1] - ourbins[:, 0])
p2 = plt.bar(x = ourbins[:,0], height = n_norm[:,1],width = ourbins[:, 1] - ourbins[:, 0],bottom = n_no
rm[:,0])
plt.legend(['Response = Yes', 'Response = No'])
plt.title(''Normalized Histogram of Age with Response Overlay'')
plt.xlabel(''Age'')
plt.ylabel(''Proportion'')
plt.show()
```



Strength: Shows relationship between target and predictor variable Weakness: Does not show distribution of predictor variable

Chapter 6

Questions #14, 15, 16, & 17 adult_ch6_training and adult_ch6_test data sets

In [23]:
```
#load in data sets
train = pd.read_csv("C:/Users/Filipp/Documents/usd_data_sci/502_data mining/module1/Website Data Sets/a
dult_ch6_training")
test = pd.read_csv("C:/Users/Filipp/Documents/usd_data_sci/502_data mining/module1/Website Data Sets/ad
ult_ch6_test")
```

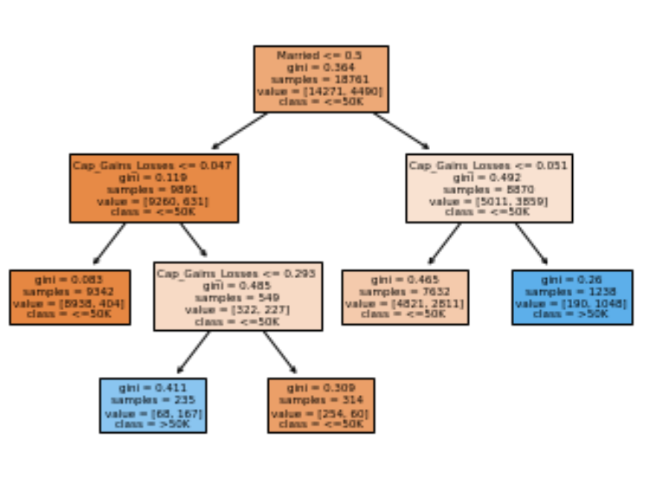
Question 14

Create a CART model using the training data set that predicts income using marital status and capital gains and losses. Visualize the decision tree (that is, provide the decision tree output). Describe the first few splits in the decision tree.

In [41]:
```
import pandas as pd
import numpy as np
import statsmodels.tools.tools as stattools
import sklearn.tree as tree
from sklearn.tree import DecisionTreeClassifier, export_graphviz
```

In [55]:
```
adult_tr = train
y = adult_tr[['Income']]
mar_np = np.array(adult_tr['Marital status'])
(mar_cat, mar_cat_dict) = stattools.categorical(mar_np,drop=True, dictnames = True)

mar_cat_pd = pd.DataFrame(mar_cat)
X = pd.concat((adult_tr[['Cap_Gains_Losses']], mar_cat_pd), axis = 1)

X_names = ["Cap_Gains_Losses", "Divorced", "Married","Never-married","Separated", "Widowed"]

y_names = ["<=50K", ">50K"]

cart01 = DecisionTreeClassifier(criterion = "gini", max_leaf_nodes=5).fit(X,y);
```

In [56]:
```
tree.plot_tree(cart01,feature_names=X_names,class_names=y_names,filled="true");
```



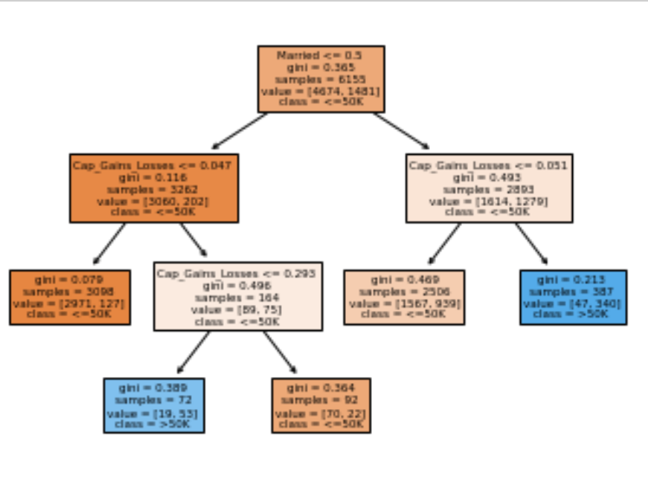
From the root node, we can confirm that about 24% of our dataset has an income of >50k. Those are not married comprise about 53% of our dataaset, and 6% of those individuals have an income of >50k. Similarly, Married people comprise 47% of our data, and 47% of those individuals have an income >50k. Non-married individuals with a capital gains loss of less than 4.7% make up 50% of our total data, and 4% of this subgroup has an income of >50k, which is one of our terminal nodes.

44% of our married group has a high income, while only 6% of our non-married group has a high income (>50k).

Question 15

Develop a CART model using the test data set that utilizes the same target and predictor variables. Visualize the decision tree. Compare the decision trees. Does the test data result match the training data result

In [57]:
```
adult_tr = test
y = adult_tr[['Income']]
mar_np = np.array(adult_tr['Marital status'])
(mar_cat, mar_cat_dict) = stattools.categorical(mar_np,drop=True, dictnames = True)

mar_cat_pd = pd.DataFrame(mar_cat)
X = pd.concat((adult_tr[['Cap_Gains_Losses']], mar_cat_pd), axis = 1)

X_names = ["Cap_Gains_Losses", "Divorced", "Married","Never-married","Separated", "Widowed"]

y_names = ["<=50K", ">50K"]

cart01 = DecisionTreeClassifier(criterion = "gini", max_leaf_nodes=5).fit(X,y)
tree.plot_tree(cart01,feature_names=X_names,class_names=y_names,filled="true");
```

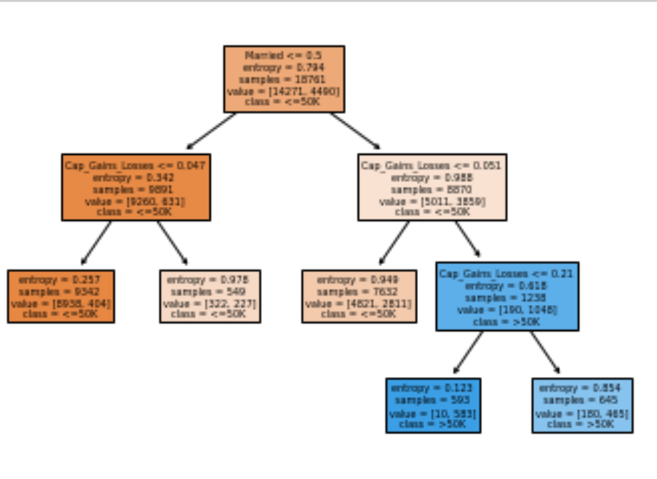


Yes, the decision trees are similar - the only discrepancy occurs at the terminal nodes for non-married instances with a capital gains loss greater than or equal to 4.7% and less than 29%, wherein the proportion of high income individuals is 5 percent greater than in the training set.

Question 16

Use the training data set to build a C5.0 model to predict income using marital status and capital gains and losses. Specify a minimum of 75 cases per terminal node. Visualize the decision tree. Describe the first few splits in the decision tree.

In [54]:
```
import warnings
warnings.filterwarnings('ignore')
adult_tr = train
y = adult_tr[['Income']]
mar_np = np.array(adult_tr['Marital status'])
(mar_cat, mar_cat_dict) = stattools.categorical(mar_np,drop=True, dictnames = True)

mar_cat_pd = pd.DataFrame(mar_cat)
X = pd.concat((adult_tr[['Cap_Gains_Losses']], mar_cat_pd), axis = 1)

X_names = ["Cap_Gains_Losses", "Divorced", "Married","Never-married","Separated", "Widowed"]

y_names = ["<=50K", ">50K"]
c50_01 = DecisionTreeClassifier(criterion="entropy",max_leaf_nodes=5).fit(X,y)
tree.plot_tree(c50_01,feature_names=X_names,class_names=y_names,filled="true");
```



Similarities: similar splitting along leftmost node for CGL with a cutoff of 0.047. Same amount of levels. Leftmost node bins most of the dataset after only assessing CGL, which also happens in our CART model.

Differences: Third level split for CGL happens for different values of marriage, model looks at entropy instead of gini for splitting.

Chapter 11 Questions 34-41

For the following exercises, work with the bank_reg_training and the bank_reg_test data sets. Use either Python or R to solve each problem.

In [59]:
```
#load in datasets
test = pd.read_csv("C:/Users/Filipp/Documents/usd_data_sci/502_data mining/module1/Website Data Sets/ba
nk_reg_training")
train= pd.read_csv("C:/Users/Filipp/Documents/usd_data_sci/502_data mining/module1/Website Data Sets/ba
nk_reg_test")
```

Question 34

Use the training set to run a regression predicting Credit Score, based on Debt-to-Income Ratio and Request Amount. Obtain a summary of the model. Do both predictors belong in the model?

In [61]:
```
import statsmodels.api as sm
train.head()
```

Out[61]:

	Approval	Credit Score	Debt-to-Income Ratio	Interest	Request Amount
0	T	767.0	0.05	2700.0	6000.0
1	F	707.0	0.05	12150.0	27000.0
2	T	664.0	0.08	900.0	2000.0
3	F	652.0	0.05	9000.0	20000.0
4	T	664.0	0.27	5850.0	13000.0

In [62]:
```
X = pd.DataFrame(train[['Request Amount','Debt-to-Income Ratio']])
y = pd.DataFrame(train[['Credit Score']])
```

In [63]:
```
X = sm.add_constant(X)
```

In [64]:
```
model01 = sm.OLS(y, X).fit()
```

In [65]:
```
model01.summary()
```

Out[65]:

OLS Regression Results

Dep. Variable:	Credit Score	**R-squared:**	0.038
Model:	OLS	**Adj. R-squared:**	0.038
Method:	Least Squares	**F-statistic:**	215.4
Date:	Tue, 16 Mar 2021	**Prob (F-statistic):**	1.94e-92
Time:	01:23:46	**Log-Likelihood:**	-60395.
No. Observations:	10775	**AIC:**	1.208e+05
Df Residuals:	10772	**BIC:**	1.208e+05
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	665.4987	1.328	501.265	0.000	662.896	668.101
Request Amount	0.0013	6.85e-05	19.013	0.000	0.001	0.001
Debt-to-Income Ratio	-52.1374	4.826	-10.803	0.000	-61.597	-42.677

Omnibus:	1792.693	**Durbin-Watson:**	1.985
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	3194.120
Skew:	-1.067	**Prob(JB):**	0.00
Kurtosis:	4.600	**Cond. No.**	1.25e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.25e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

Both predictors are significant.

1. Validate the model from the previous exercise.

In [67]:
```
X = pd.DataFrame(test[['Request Amount','Debt-to-Income Ratio']])
y = pd.DataFrame(test[['Credit Score']])
X = sm.add_constant(X)
validate_01 = sm.OLS(y, X).fit()
validate_01.summary()
```

Out[67]:

OLS Regression Results

Dep. Variable:	Credit Score	**R-squared:**	0.028
Model:	OLS	**Adj. R-squared:**	0.028
Method:	Least Squares	**F-statistic:**	156.2
Date:	Tue, 16 Mar 2021	**Prob (F-statistic):**	1.37e-67
Time:	01:26:12	**Log-Likelihood:**	-59972.
No. Observations:	10693	**AIC:**	1.199e+05
Df Residuals:	10690	**BIC:**	1.200e+05
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	668.4562	1.336	500.275	0.000	665.837	671.075
Request Amount	0.0011	6.84e-05	15.727	0.000	0.001	0.001
Debt-to-Income Ratio	-48.1262	4.785	-10.058	0.000	-57.505	-38.747

Omnibus:	1658.575	**Durbin-Watson:**	1.991
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	2844.250
Skew:	-1.021	**Prob(JB):**	0.00
Kurtosis:	4.487	**Cond. No.**	1.24e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.24e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

Validation holds up - we have similar coefficients and all variables are significant.

1. Use the regression equation to complete this sentence: “The estimated Credit Score equals….” **668 plus 0.001 times the request amount (in dollars) minus 48 points times the debt to income ratio**

1. Interpret the coefficient for Debt-to-Income Ratio.
   a one-unit increase in the debt to income ratio, that is, as an individual takes on more debt relative to their income, are associated with a 48-point drop in their credit score.

1. Interpret the coefficient for Request Amount.

A one-unit increase in the request amount is associated with a 0.001 point increase in the credit score.

1. Find and interpret the value of s.

In [70]:
```
np.sqrt(validate_01.scale)
```

Out[70]:
```
66.00195259717188
```

Our standard error is about 66 points, which means that the point estimate for any given object can be expected to have an error of roughly 66 credit score points.

1. Find and interpret adjusted R^2. Comment.

The R-squared value is 0.028, or about 3 percent, suggesting very poor model performance. We would not expect this model to do notably better than a baseline model.

1. Find MAEBaseline and MAERegression, and determine whether the regression model outperformed its baseline model.

In [75]:
```
#MAE REGRESSION
import sklearn.metrics as met
X_test = pd.DataFrame(test[['Request Amount','Debt-to-Income Ratio']])
X_test = sm.add_constant(X_test)
ypred = validate_01.predict(X_test)
ytrue = test[['Credit Score']]
met.mean_absolute_error(y_true = ytrue, y_pred = ypred)
```

Out[75]:
```
48.309615690591095
```

In [84]:
```
#MAE BASELINE
x = np.abs(ytrue - np.mean(ytrue))
np.sum(x)/len(x)
```

Out[84]:
```
Credit Score    48.709696
dtype: float64
```

Our baseline MAE is almost inseparable from our model MAE, suggesting that we would be indifferent to either of these in a practical environment.

In [ ]: