2. Problem Statement In this assignment students need to predict whether a person makes over 50K per year or not from classic adult dataset using XGBoost. The description of the dataset is as follows: Data Set Information: Extraction was done by Barry Becker from the 1994 Census database. A set of reasonably clean records was extracted using the following conditions: ((AAGE>16) && (AGI>100) && (AFNLWGT>1)&& (HRSWK>0)) Attribute Information: Listing of attributes: >50K, <=50K. age: continuous. workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked. fnlwgt: continuous. education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool. education-num: continuous. marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse. occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Profspecialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces. relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried. race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black. sex: Female, Male. capital-gain: continuous. capital-loss: continuous. hours-per-week: continuous. native-country: 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. Following is the code to load required libraries and data: import numpy as np import pandas as pd train\_set = pd.read\_csv('http://archive.ics.uci.edu/ml/machine-learningdatabases/adult/adult.data', header = None) test\_set = pd.read\_csv('http://archive.ics.uci.edu/ml/machine-learningdatabases/adult/adult.test', skiprows = 1, header = None) col\_labels = ['age', 'workclass', 'fnlwgt', 'education', 'education\_num', 'marital\_status', 'occupation','relationship', 'race', 'sex', capital\_gain', 'capital\_loss', 'hours\_per\_week', 'native\_country', 'wage\_class'] train\_set.columns = col\_labels test\_set.columns = col\_labels NOTE: The solution shared through Github should contain the source code used and the screenshot of the output.

Answer:

**import** **pandas** **as** **pd** **import** **numpy** **as** **np** **import** **math** **import** **statsmodels** **as** **sm** **import** **sklearn** **as** **skl** **from** **sklearn.preprocessing** **import** StandardScaler **import** **sklearn.linear\_model** **as** **linear\_model** **from** **sklearn.metrics** **import** classification\_report **from** **sklearn.metrics** **import** roc\_auc\_score **import** **sklearn.metrics** **as** **metrics** **import** **sklearn.tree** **as** **tree** **import** **seaborn** **as** **sns** **import** **matplotlib.pyplot** **as** **plt** %matplotlib inline

Train\_data = pd.read\_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data', header = **None**, sep=' \*, \*', engine='python')

Test\_data = pd.read\_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.test', skiprows = 1, sep=' \*, \*', engine='python', header = **None**)

pd.set\_option('display.max\_columns', **None**)

col\_labels = ['age', 'workclass', 'fnlwgt', 'education', 'education\_num', 'marital\_status','occupation','relationship', 'race', 'sex', 'capital\_gain', 'capital\_loss', 'hours\_per\_week','native\_country', 'wage\_class']

Train\_data.columns = col\_labels

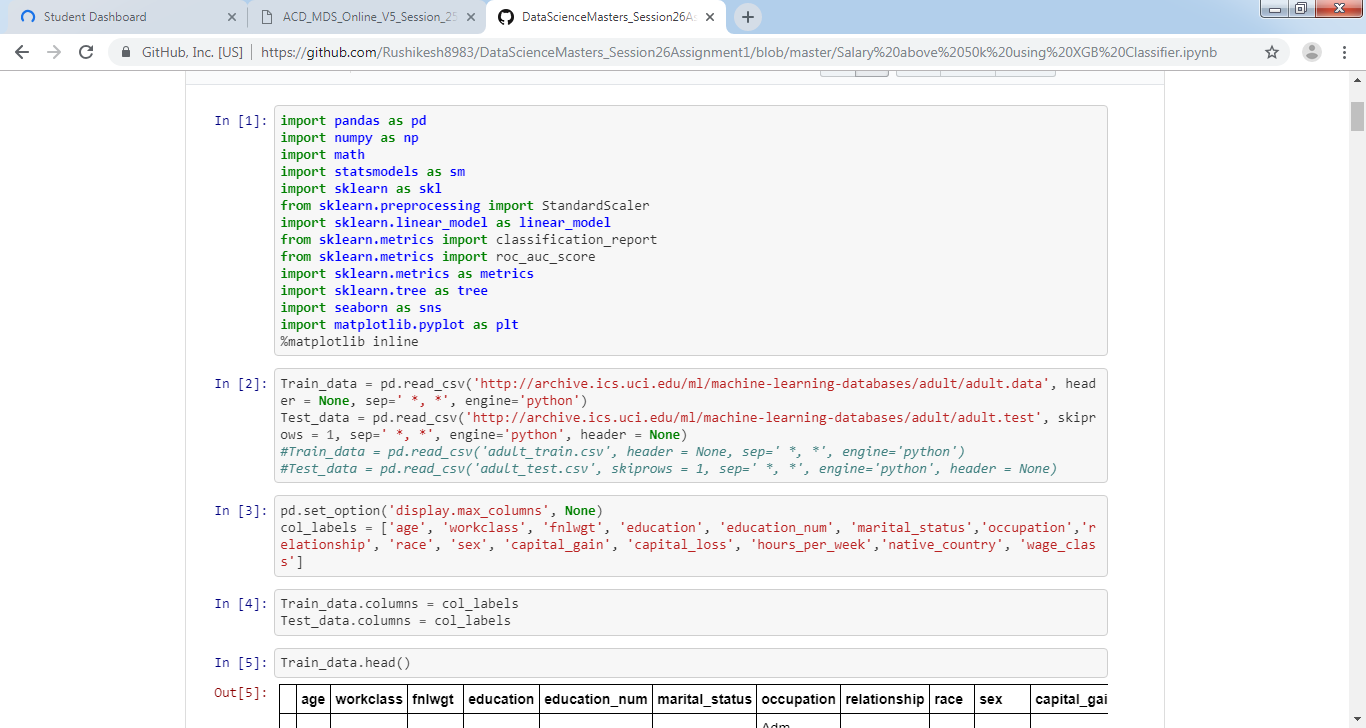
Test\_data.columns = col\_labels

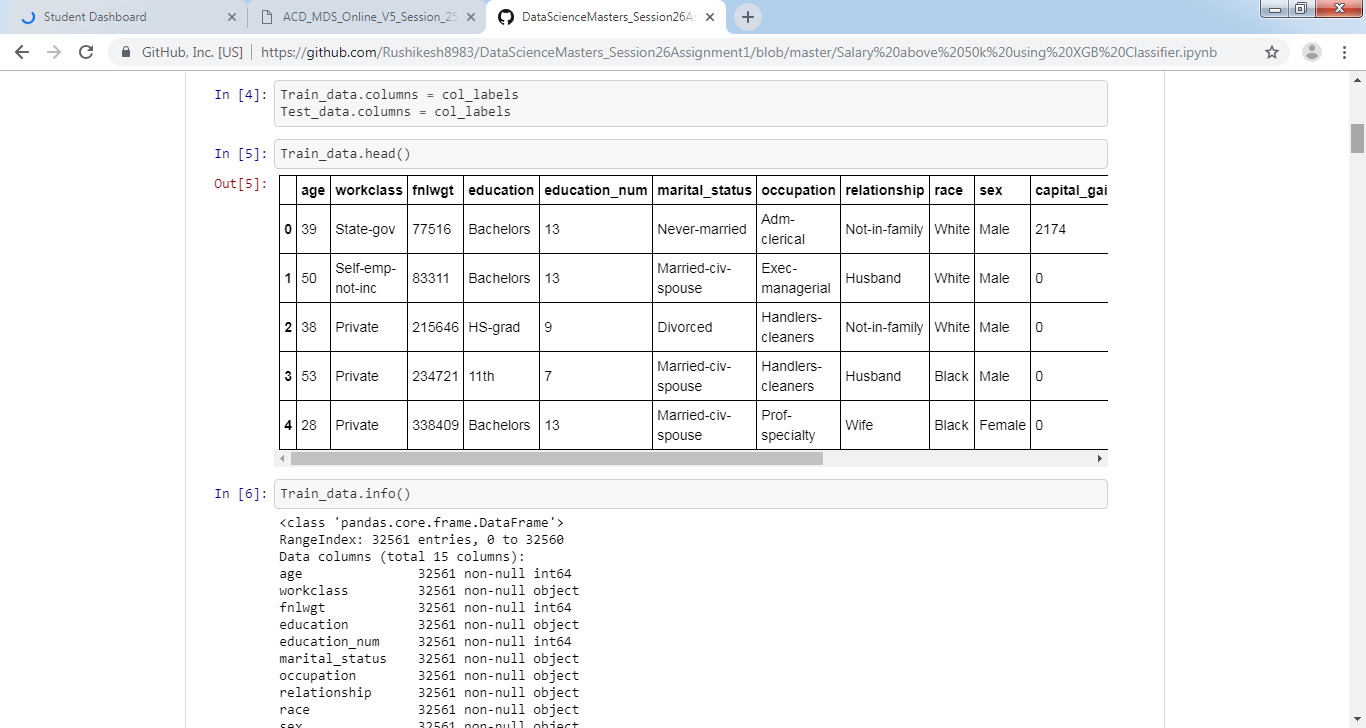
Train\_data.head()

Train\_data.info()

Test\_data.head()

Test\_data.info()





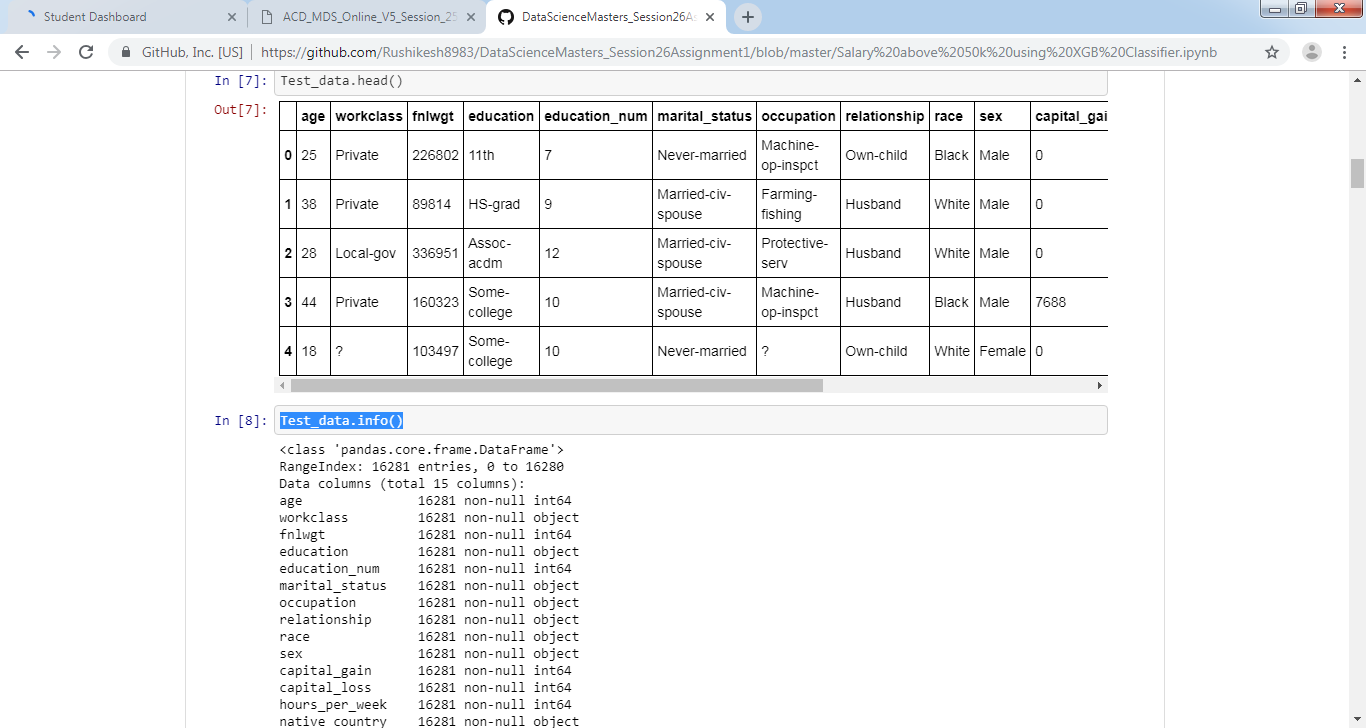


fig = plt.figure(figsize=(20,15))

cols = 5

rows = 3

**for** i, column **in** enumerate(Train\_data.columns):

ax = fig.add\_subplot(rows, cols, i + 1)

ax.set\_title(column)

**if** Train\_data.dtypes[column] == np.object:

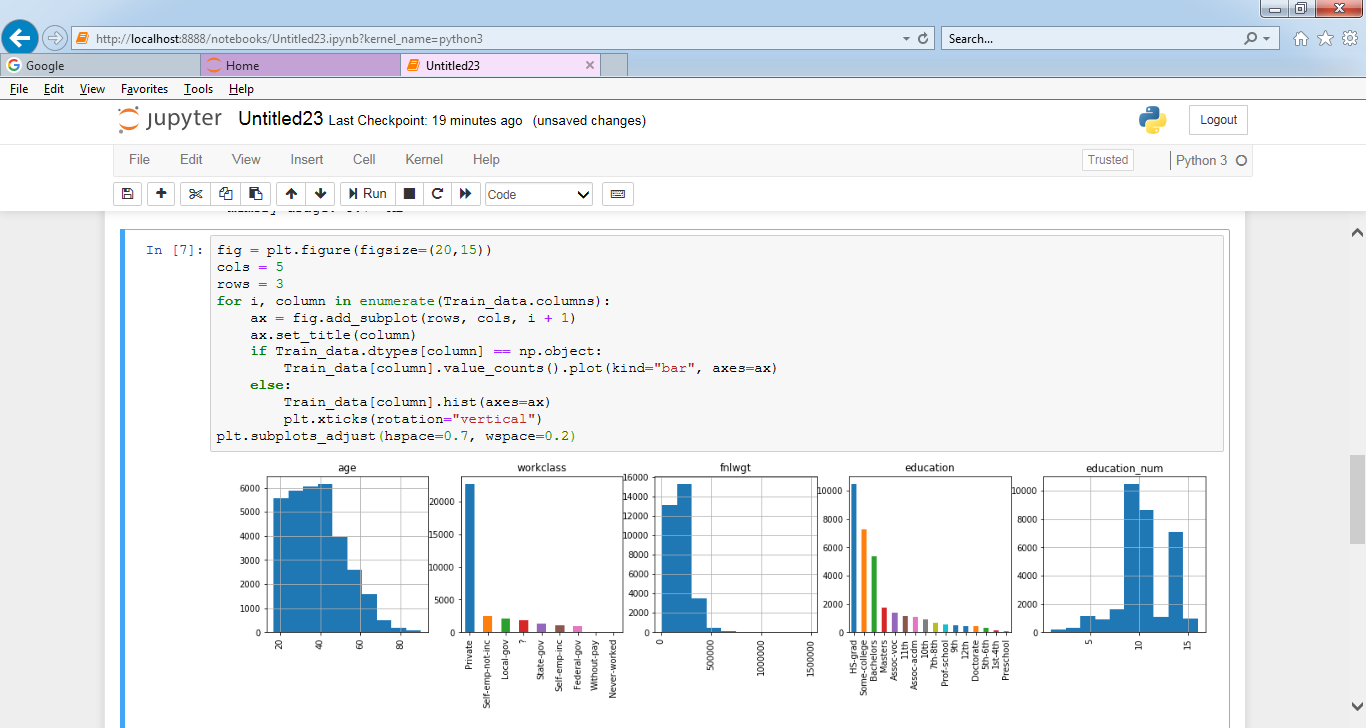
Train\_data[column].value\_counts().plot(kind="bar", axes=ax)

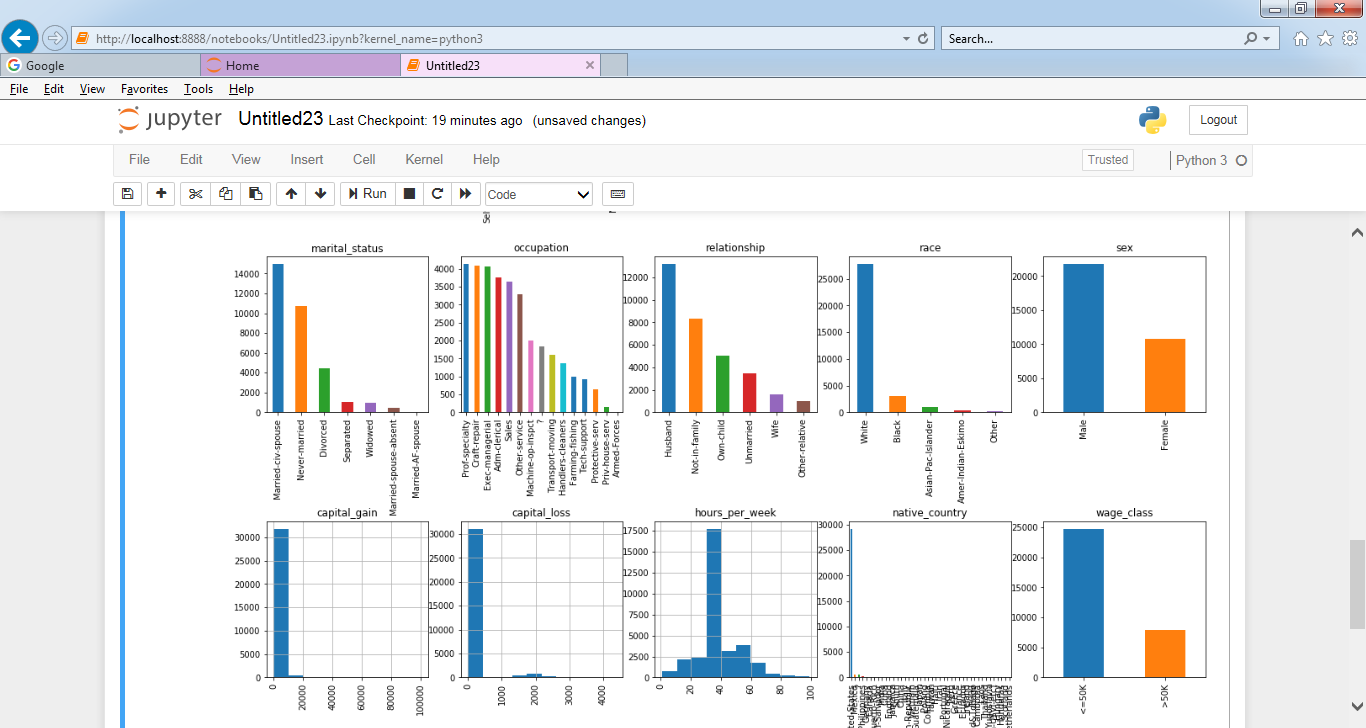
**else**:

Train\_data[column].hist(axes=ax)

plt.xticks(rotation="vertical")

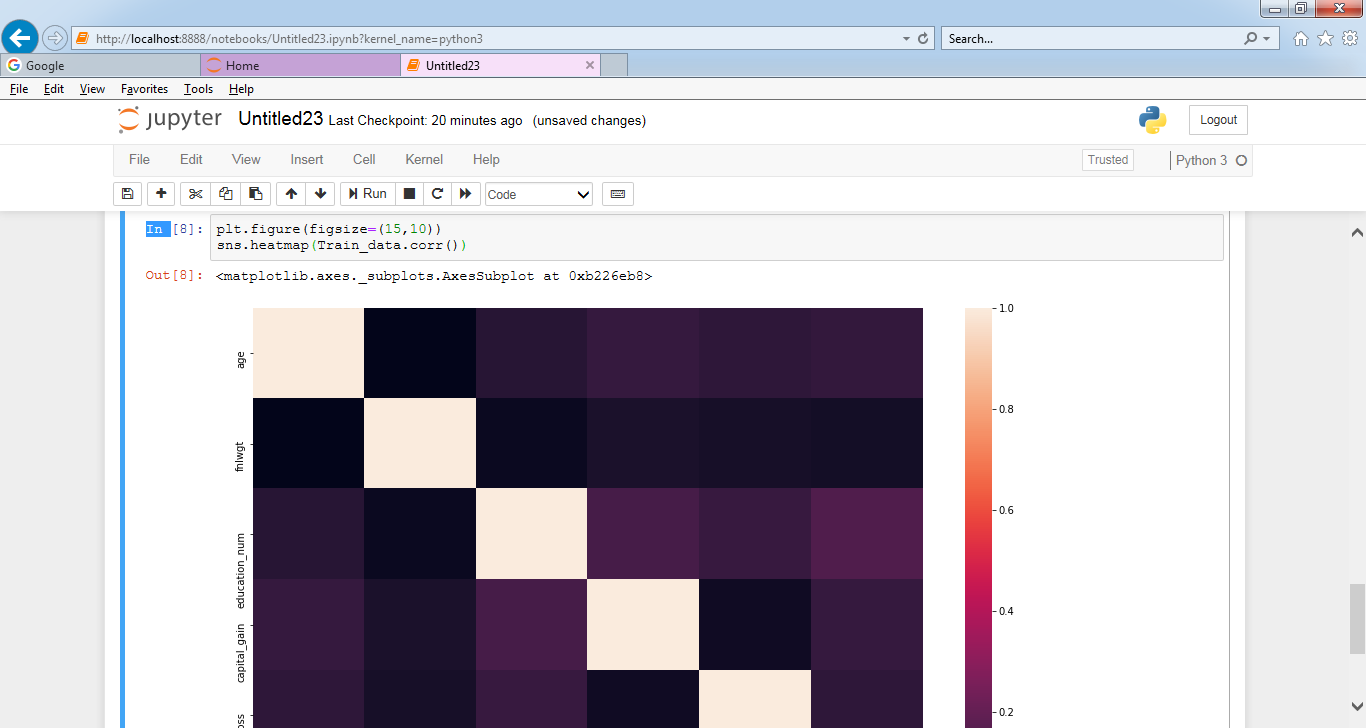
plt.subplots\_adjust(hspace=0.7, wspace=0.2)





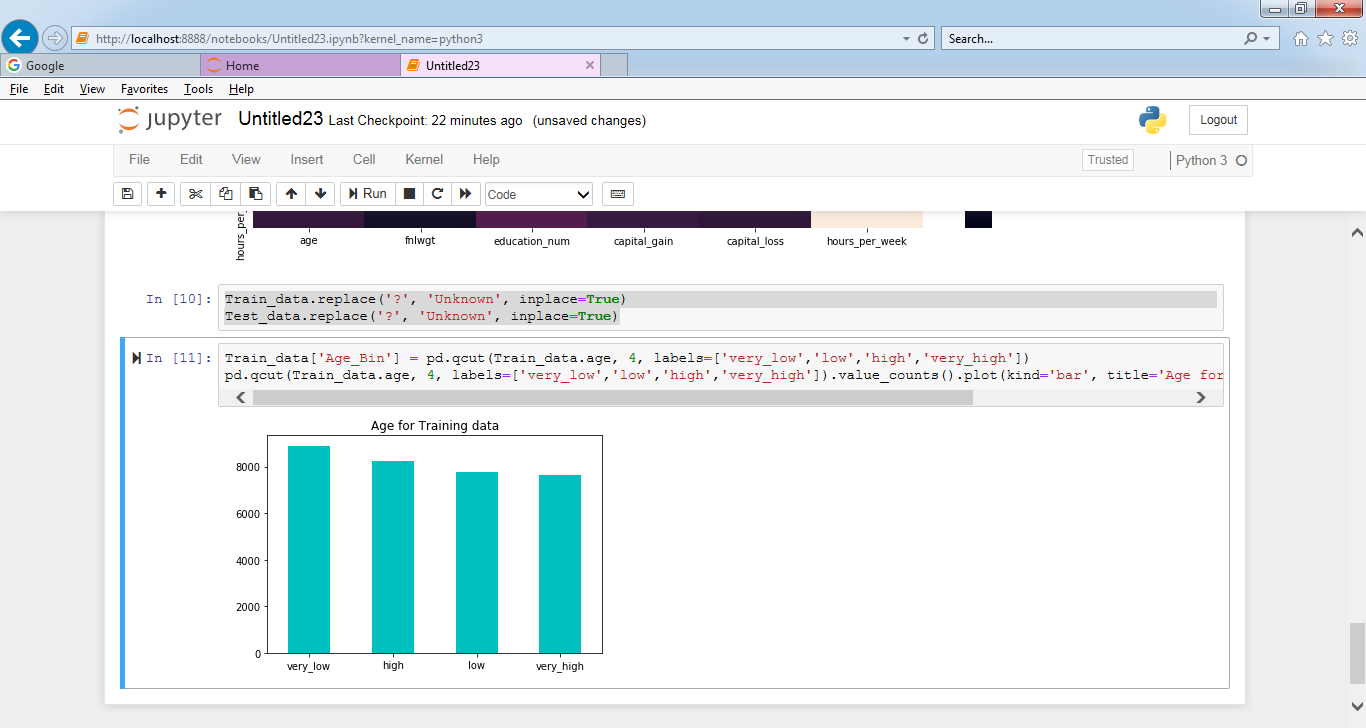
plt.figure(figsize=(15,10))

sns.heatmap(Train\_data.corr())



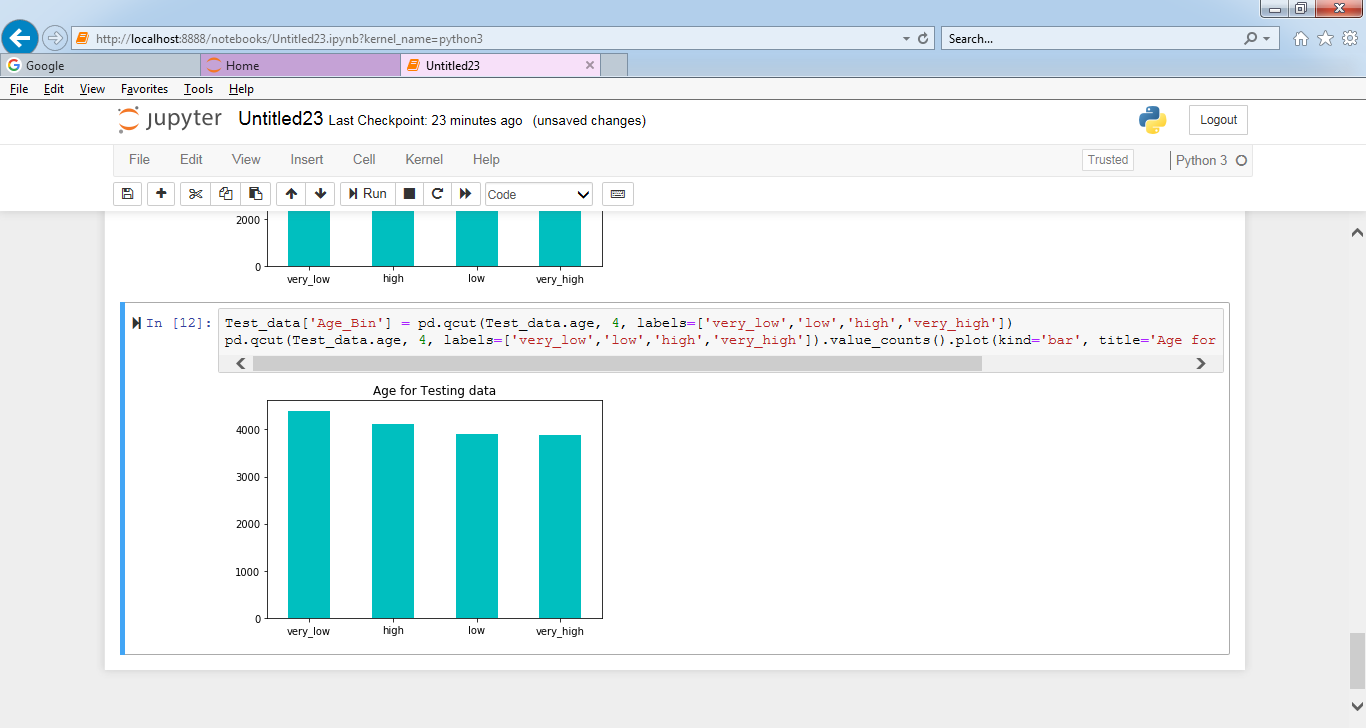
Train\_data.replace('?', 'Unknown', inplace=True)

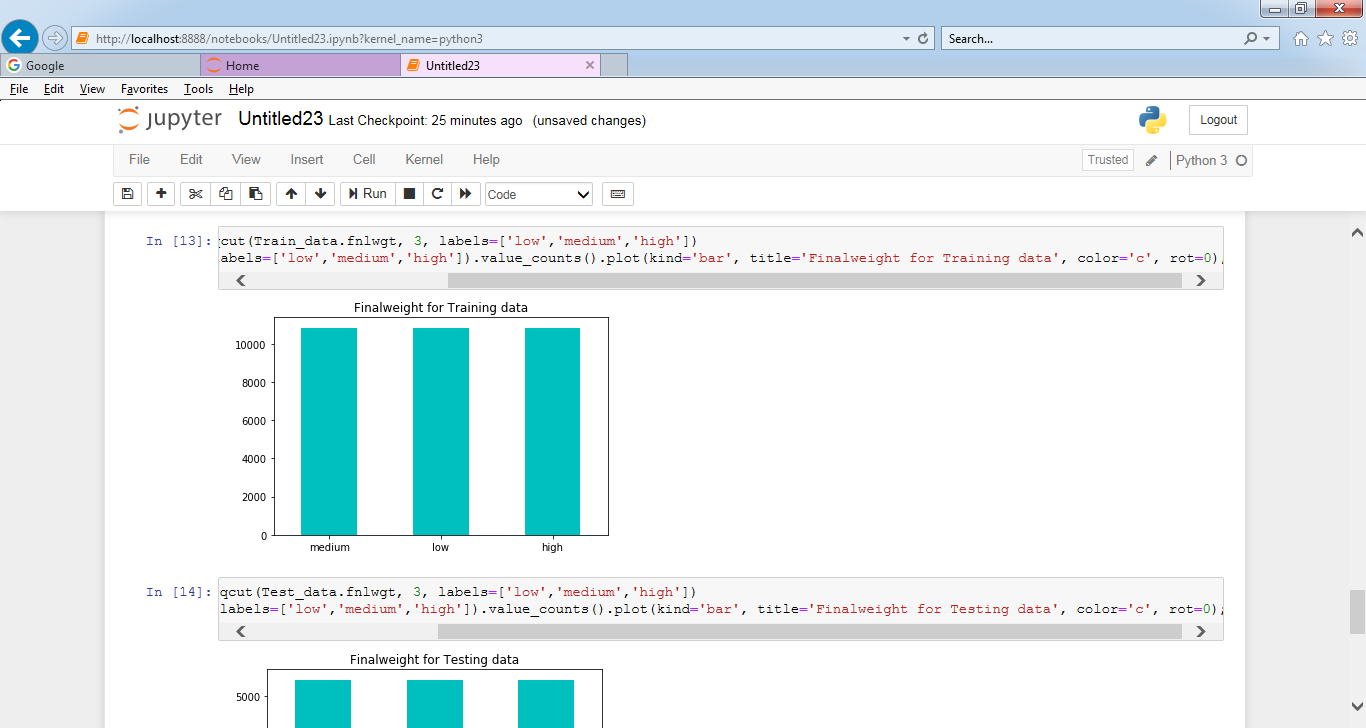
Test\_data.replace('?', 'Unknown', inplace=True)

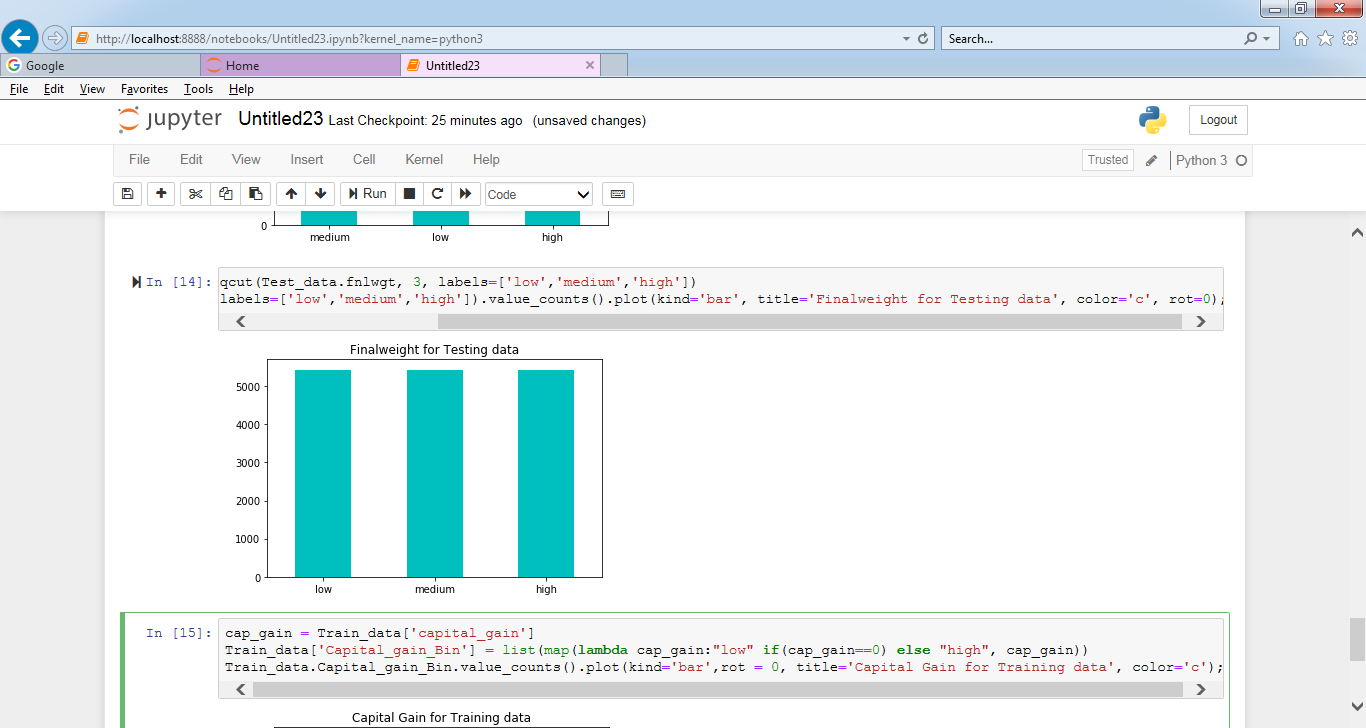


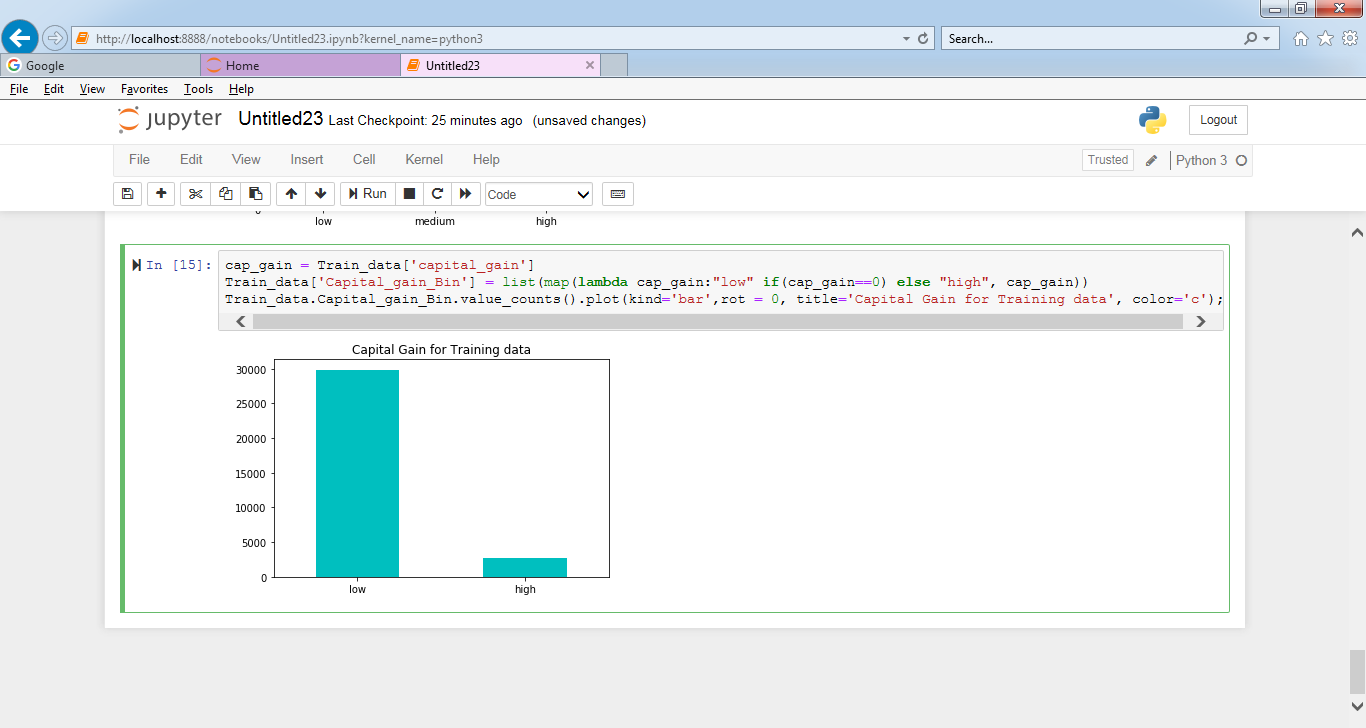
Test\_data['Age\_Bin'] = pd.qcut(Test\_data.age, 4, labels=['very\_low','low','high','very\_high'])

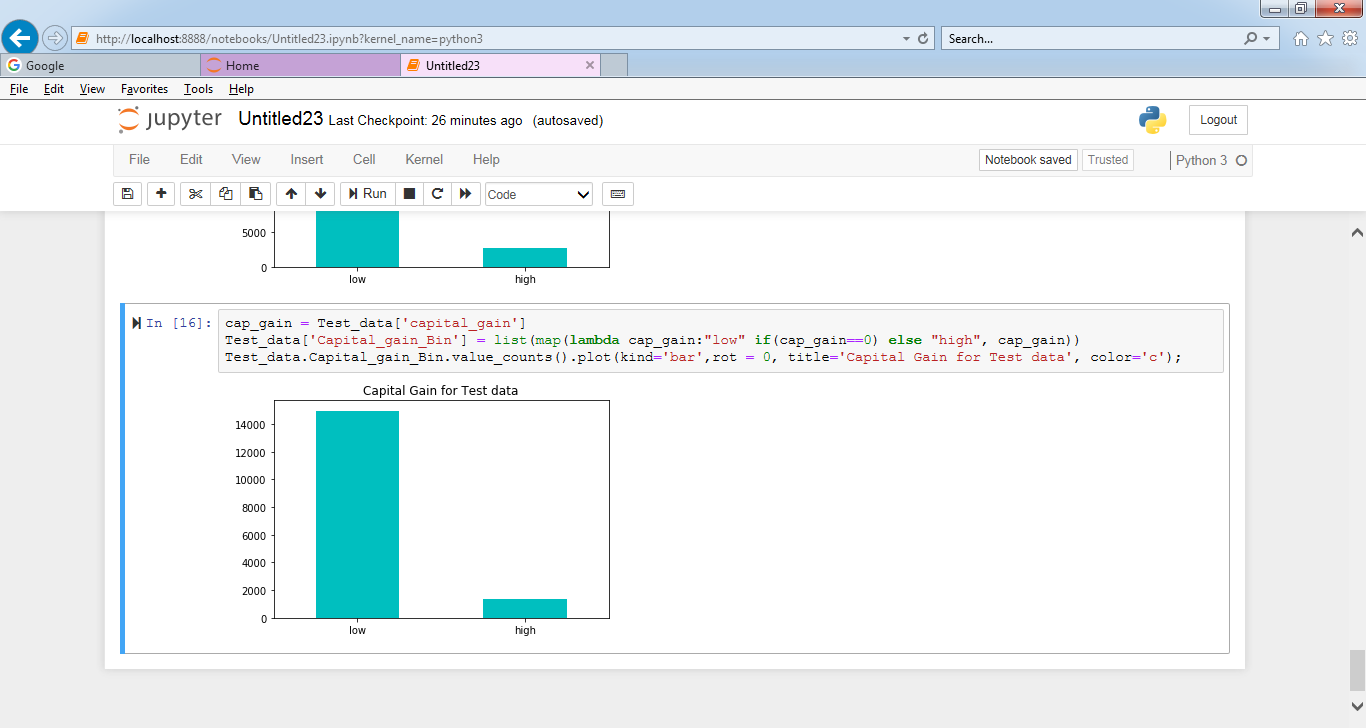
pd.qcut(Test\_data.age, 4, labels=['very\_low','low','high','very\_high']).value\_counts().plot(kind='bar', title='Age for Testing data', color='c', rot=0);

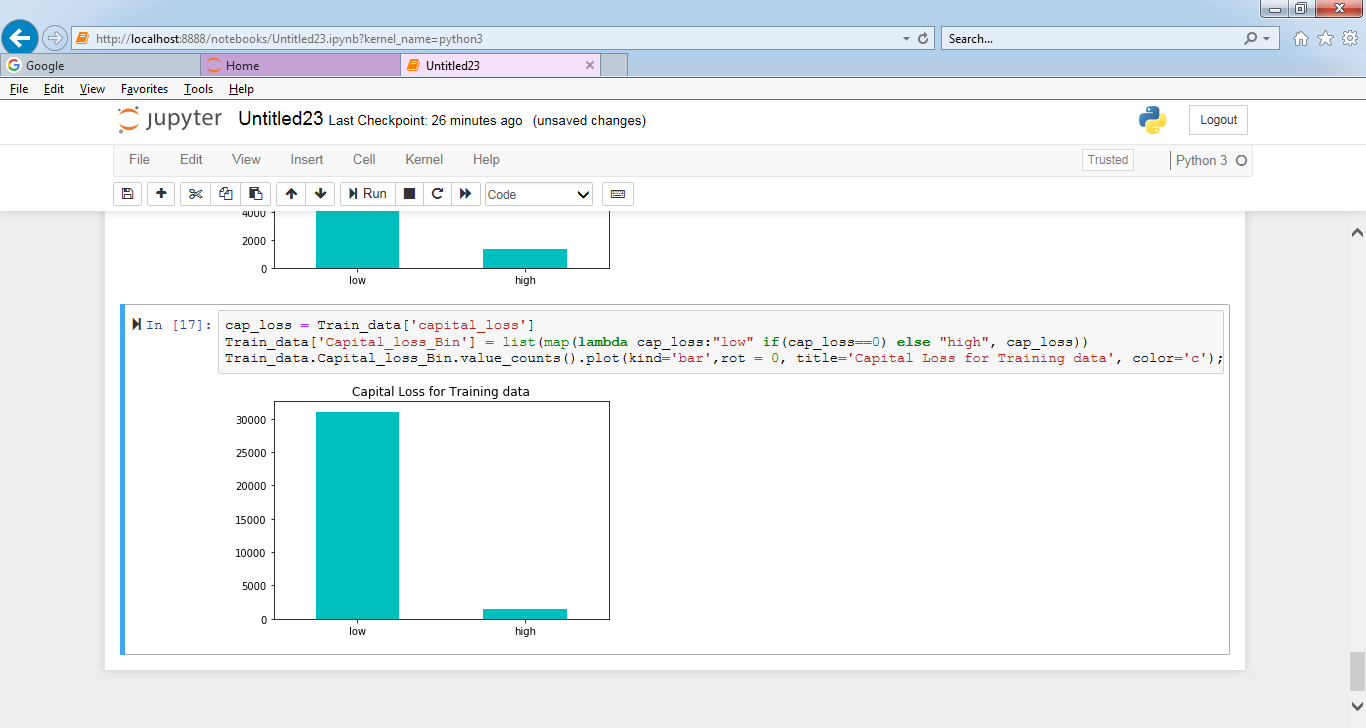


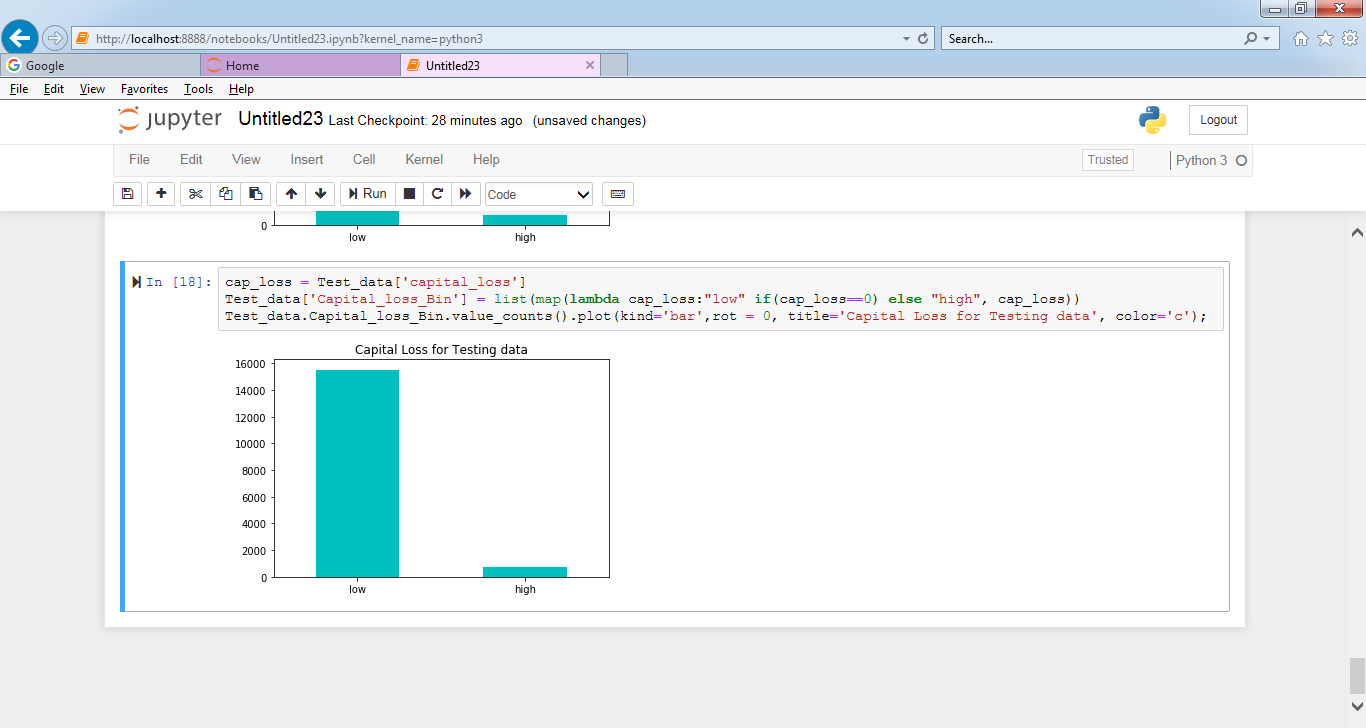








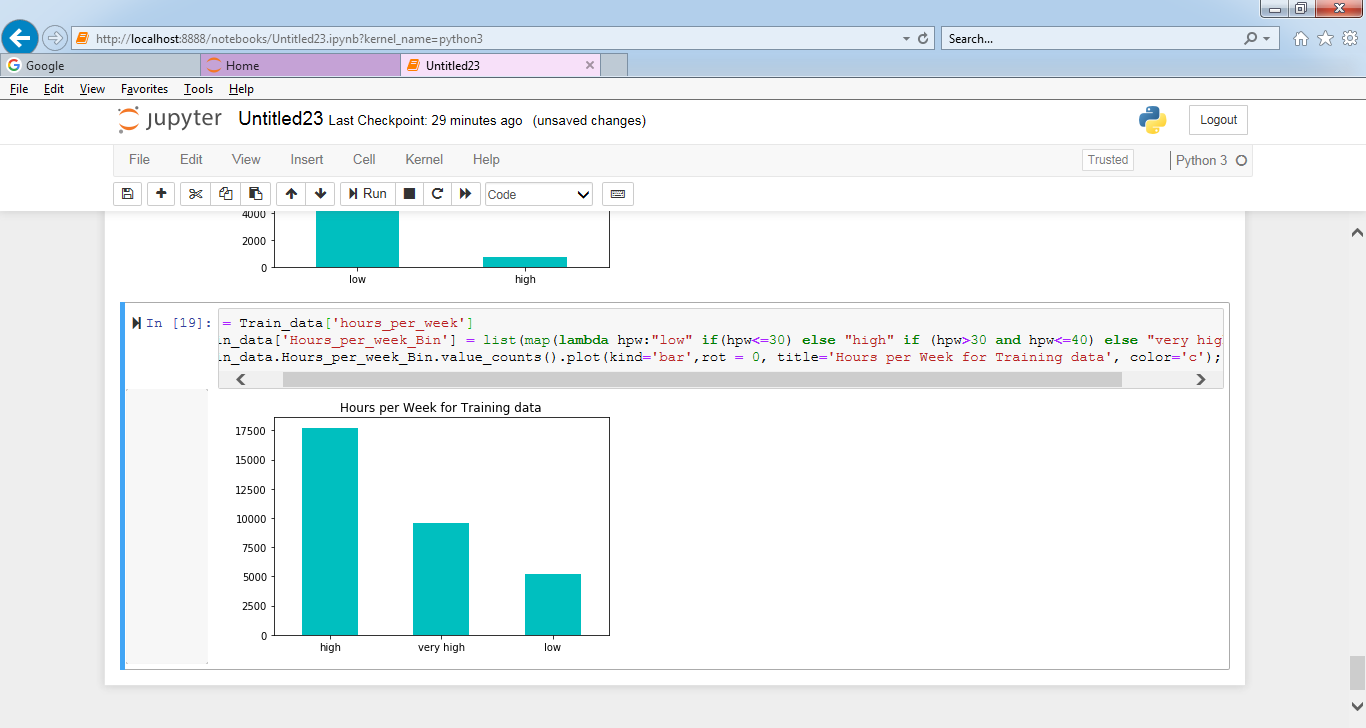




hpw = Train\_data['hours\_per\_week']

Train\_data['Hours\_per\_week\_Bin'] = list(map(**lambda** hpw:"low" **if**(hpw<=30) **else** "high" **if** (hpw>30 **and** hpw<=40) **else** "very high", hpw))

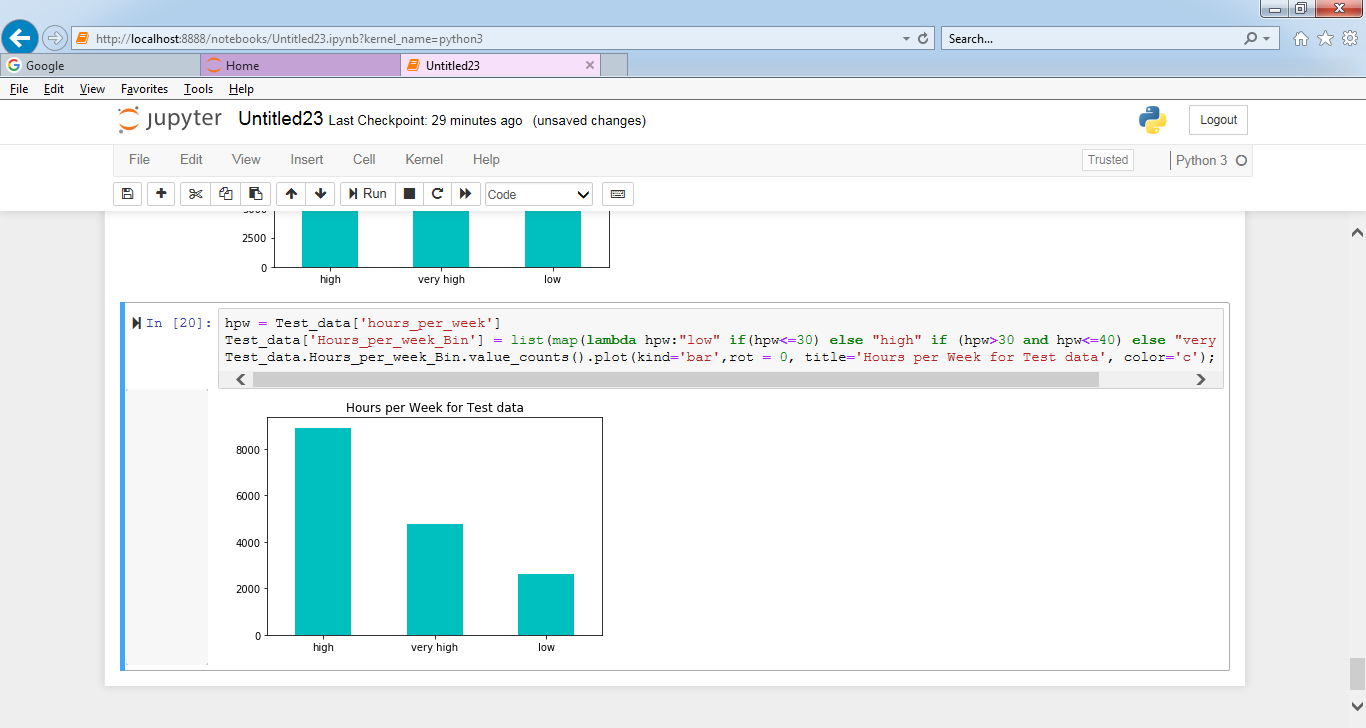
Train\_data.Hours\_per\_week\_Bin.value\_counts().plot(kind='bar',rot = 0, title='Hours per Week for Training data', color='c');



hpw = Test\_data['hours\_per\_week']

Test\_data['Hours\_per\_week\_Bin'] = list(map(**lambda** hpw:"low" **if**(hpw<=30) **else** "high" **if** (hpw>30 **and** hpw<=40) **else** "very high", hpw))

Test\_data.Hours\_per\_week\_Bin.value\_counts().plot(kind='bar',rot = 0, title='Hours per Week for Test data', color='c');



#One Hot Encoding for Feature Age, fnlwgt, capital\_gain, capital\_loss and hours\_per\_week across Training and Test dataset

Train\_dummies\_age\_bin = pd.get\_dummies(Train\_data.Age\_Bin, drop\_first=True, prefix='Age')

Train\_dummies\_fnlwgt\_bin = pd.get\_dummies(Train\_data.Fnlwgt\_Bin, drop\_first=True, prefix='Fnlwgt')

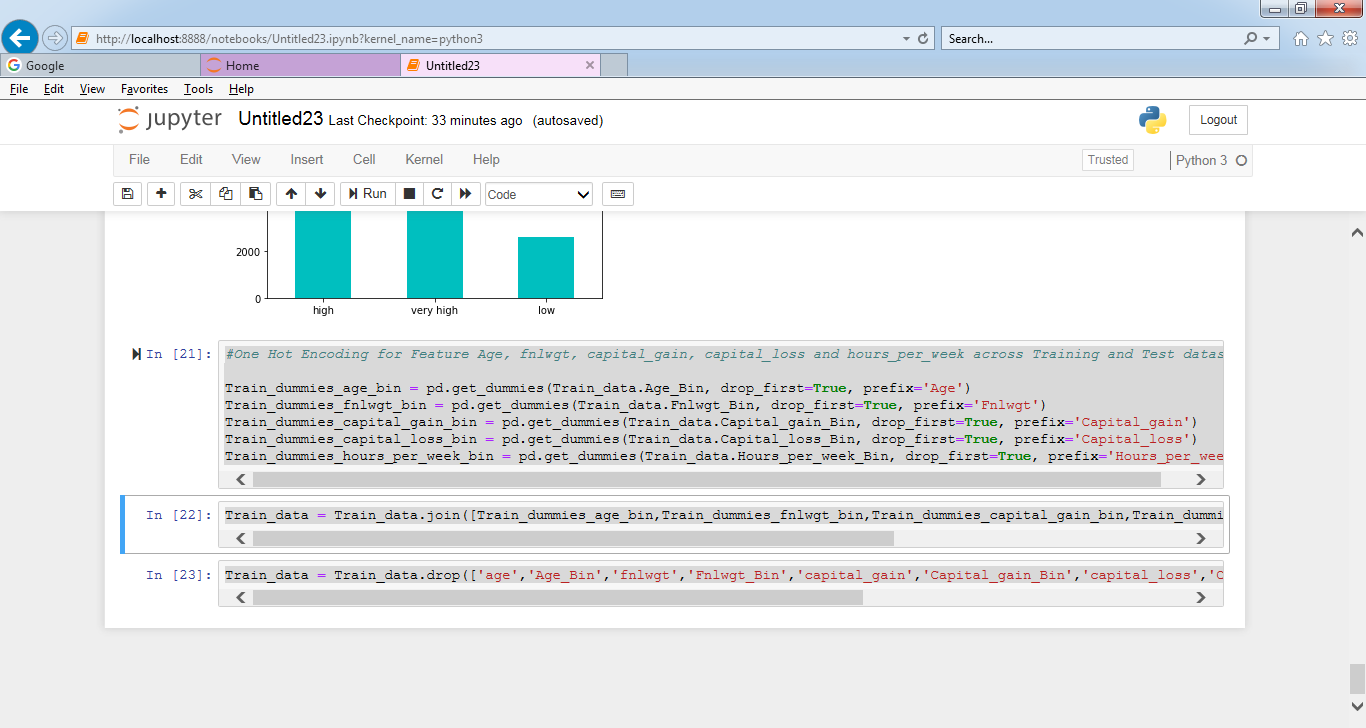
Train\_dummies\_capital\_gain\_bin = pd.get\_dummies(Train\_data.Capital\_gain\_Bin, drop\_first=True, prefix='Capital\_gain')

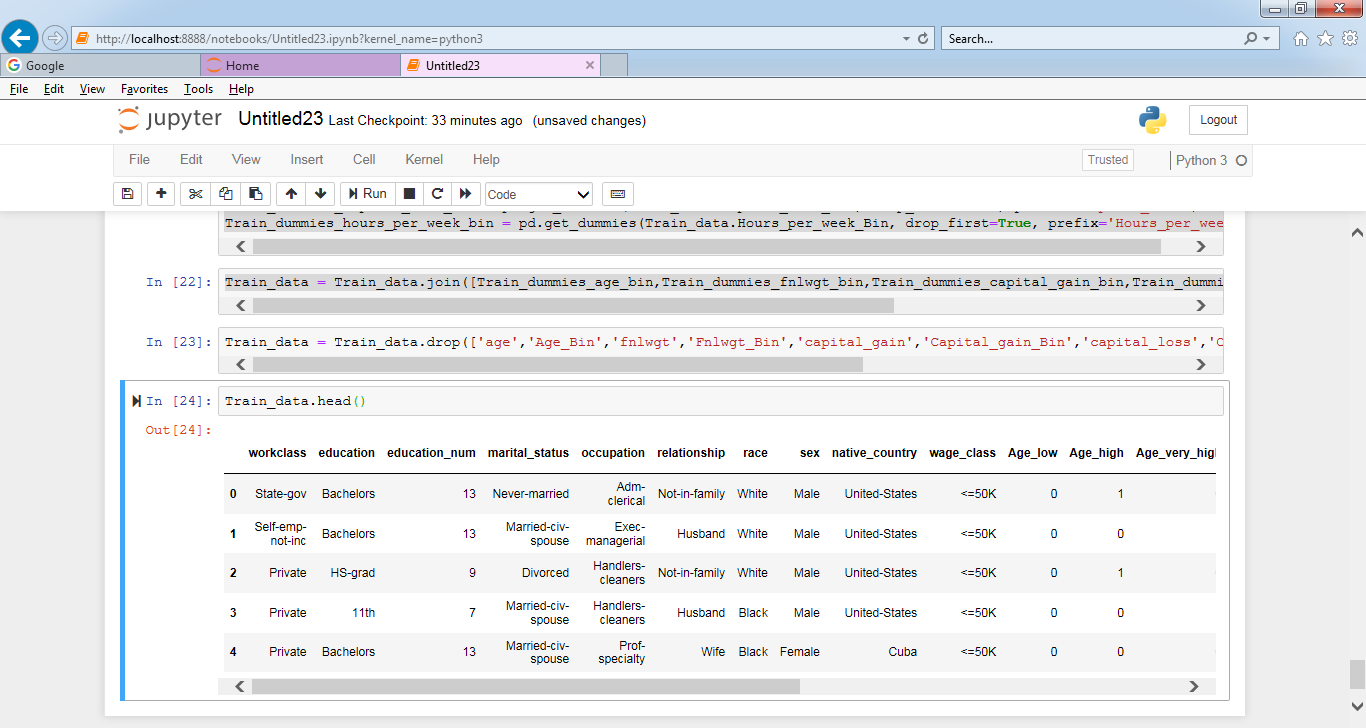
Train\_dummies\_capital\_loss\_bin = pd.get\_dummies(Train\_data.Capital\_loss\_Bin, drop\_first=True, prefix='Capital\_loss')

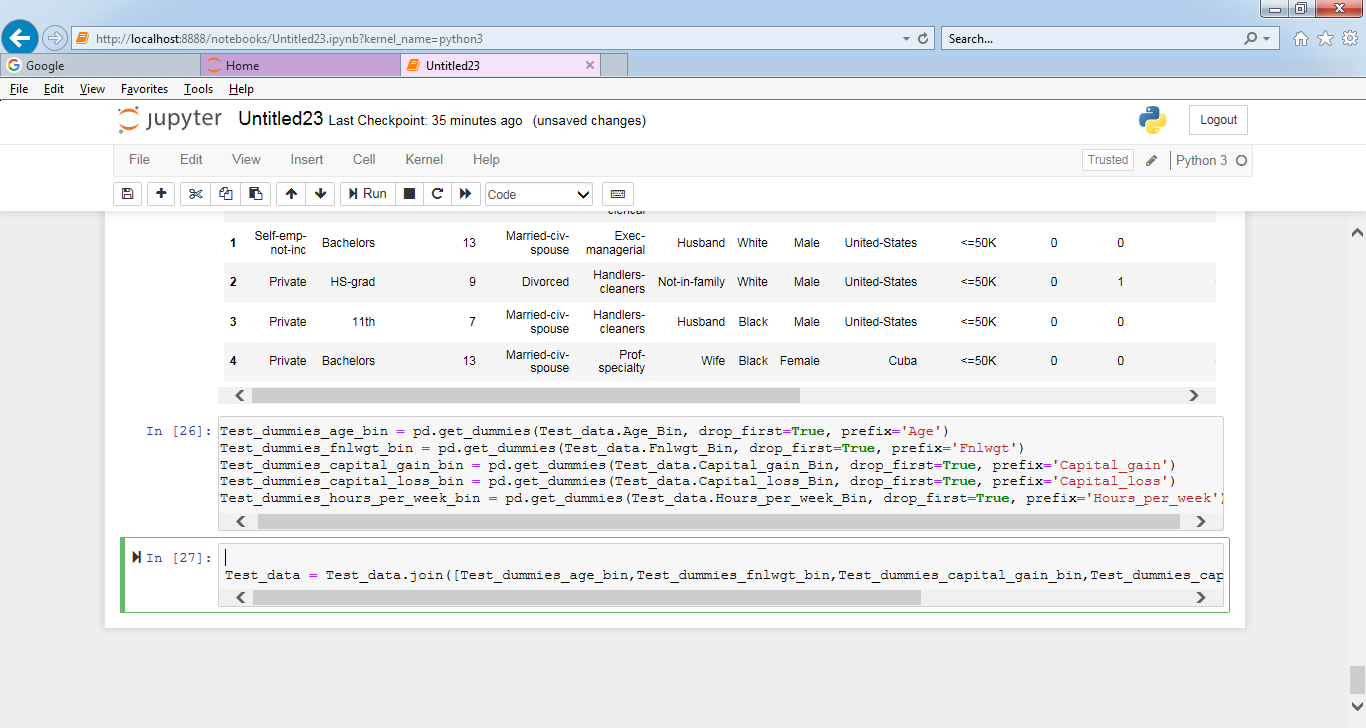
Train\_dummies\_hours\_per\_week\_bin = pd.get\_dummies(Train\_data.Hours\_per\_week\_Bin, drop\_first=True, prefix='Hours\_per\_week')

Train\_data = Train\_data.join([Train\_dummies\_age\_bin,Train\_dummies\_fnlwgt\_bin,Train\_dummies\_capital\_gain\_bin,Train\_dummies\_capital\_loss\_bin,Train\_dummies\_hours\_per\_week\_bin])

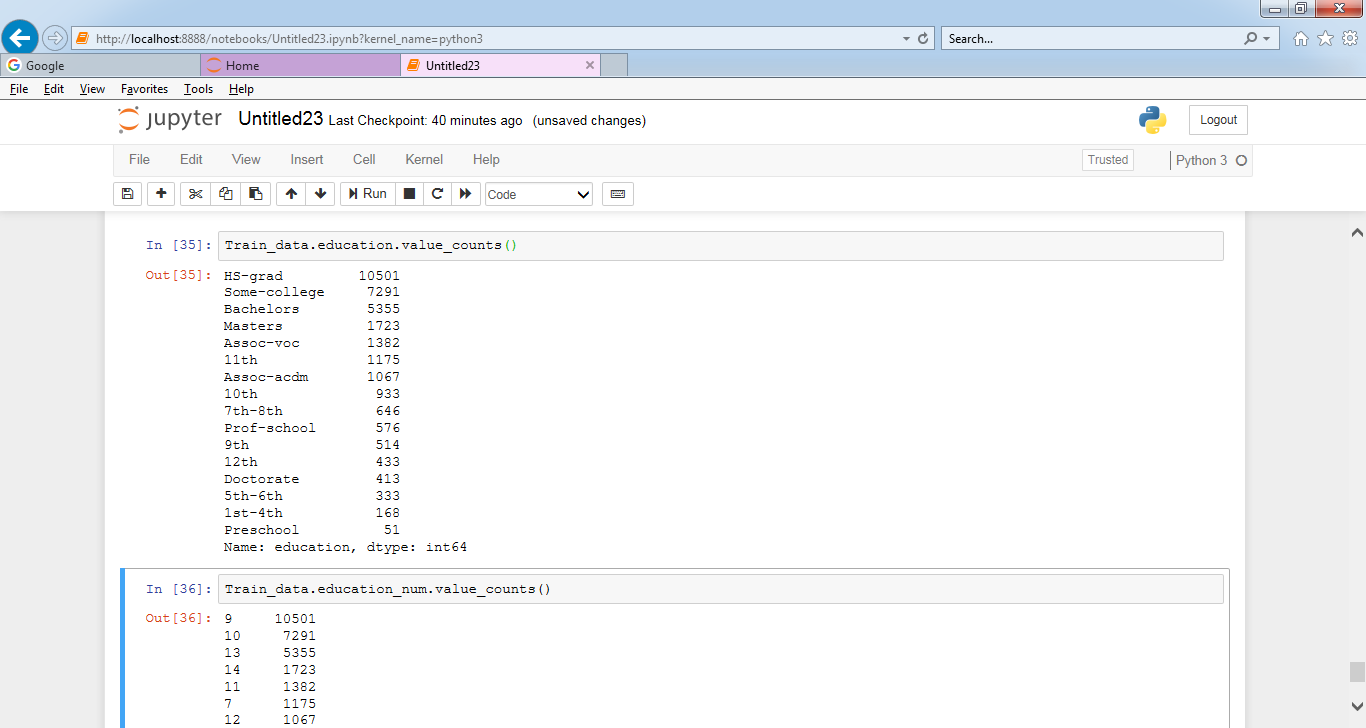
Train\_data = Train\_data.drop(['age','Age\_Bin','fnlwgt','Fnlwgt\_Bin','capital\_gain','Capital\_gain\_Bin','capital\_loss','Capital\_loss\_Bin','hours\_per\_week','Hours\_per\_week\_Bin'],axis=1)





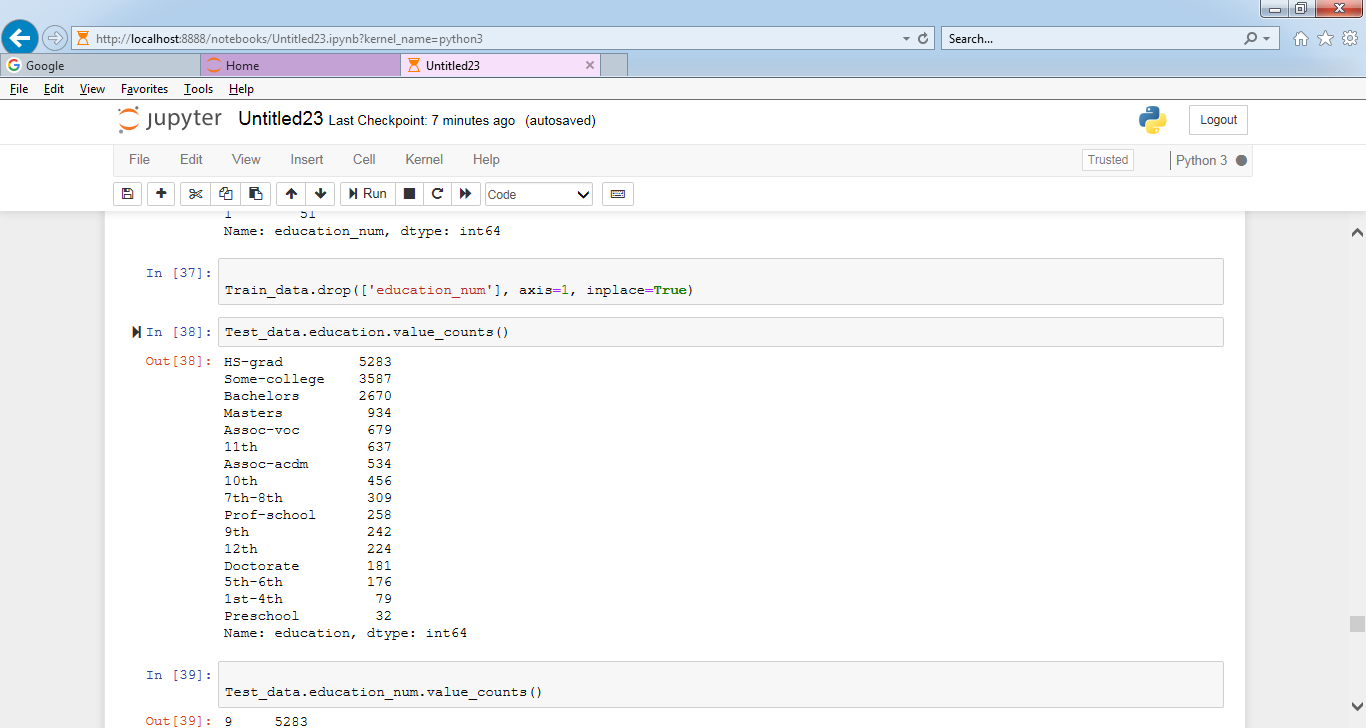


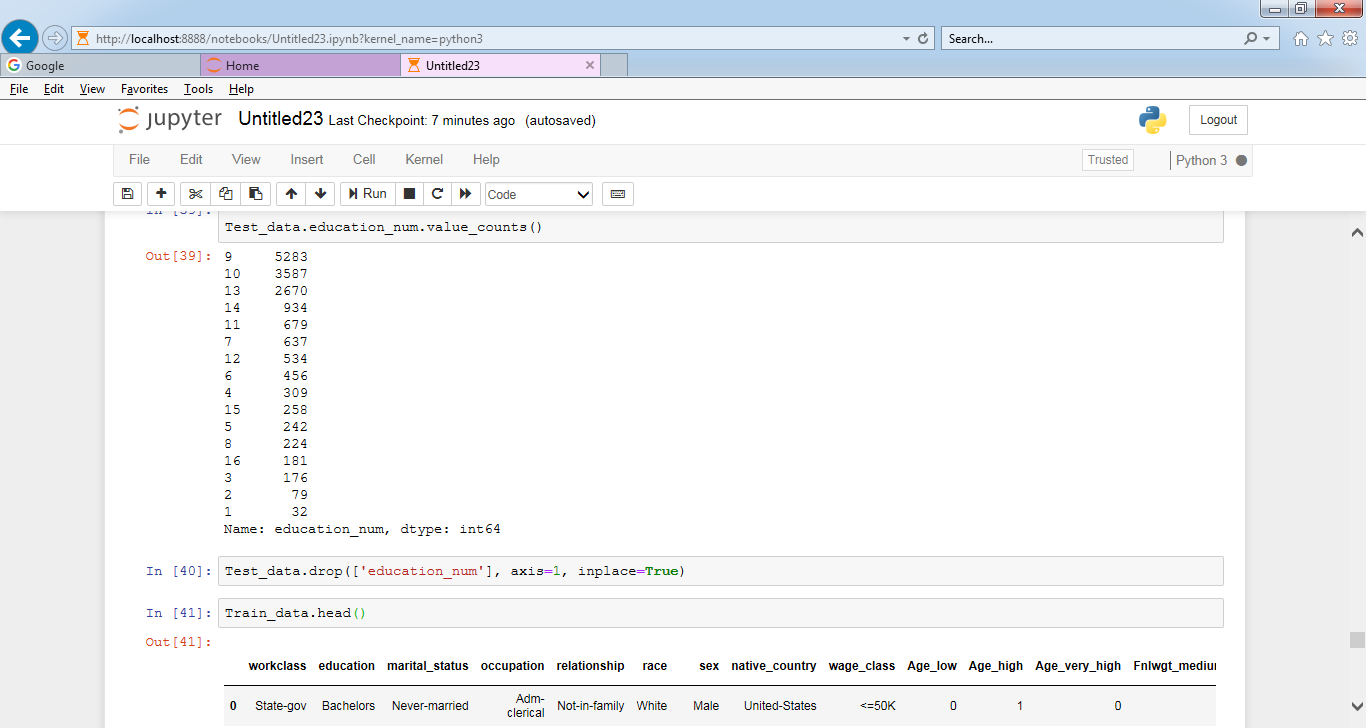
Test\_data = Test\_data.drop(['age','Age\_Bin','fnlwgt','Fnlwgt\_Bin','capital\_gain','Capital\_gain\_Bin','capital\_loss','Capital\_loss\_Bin','hours\_per\_week','Hours\_per\_week\_Bin'],axis=1)



Train\_data.drop(['education\_num'], axis=1, inplace=**True**)

Test\_data.education\_num.value\_counts()





Train\_data = pd.get\_dummies(Train\_data,columns=['workclass','education','marital\_status','occupation','relationship','race','sex'],drop\_first=True)

Test\_data = pd.get\_dummies(Test\_data,columns=['workclass','education','marital\_status','occupation','relationship','race','sex'],drop\_first=True)

Train\_data['wage\_class'] = Train\_data['wage\_class'].map({'<=50K':0,'>50K':1})

Test\_data['wage\_class'] = Test\_data['wage\_class'].map({'<=50K.':0,'>50K.':1})

