**MACHINE LEARNING**

**DATASET:**

**Adult Income Census dataset**

**NAME: SAKTHIVEL .S**

**ROLL NO: 195214173**

**CLASS:I M.Sc-CS(A)**

**SHIFT: I**

**MACHINE LEARNING PROJECT**

In this project, I have used Linear Regression Machine Learning model for the **Adult Income Census dataset** from Kaggle website.

**PROBLEM STATEMENT:**

In this Assingment, I am working through the Income Prediction problem associated with the Adult Income Census dataset. The goal is to accurately predict whether or not someone is making more or less than $50,000 a year. While working through this problem, I am following a framework I use to attack all my machine learning problems.

This data was extracted from the [1994 Census bureau database](http://www.census.gov/en.html) by Ronny Kohavi and Barry Becker (Data Mining and Visualization, Silicon Graphics). A set of reasonably clean records was extracted using the following conditions: ((AAGE>16) && (AGI>100) && (AFNLWGT>1) && (HRSWK>0)). The prediction task is to determine whether a person makes over $50K a year.

**ML METHODOLOGY:**

Linear Regression is the methodology used for training and testing the dataset. Linear Regression is a method of modeling a target value based on independent predictors. This method is mostly used for forecasting and finding out cause and effect relationship between variables. Linear Regression techniques mostly differ based on the number of independent variables and the type of relationship between the independent and dependent variables.

**DATASET DESCRIPTION**

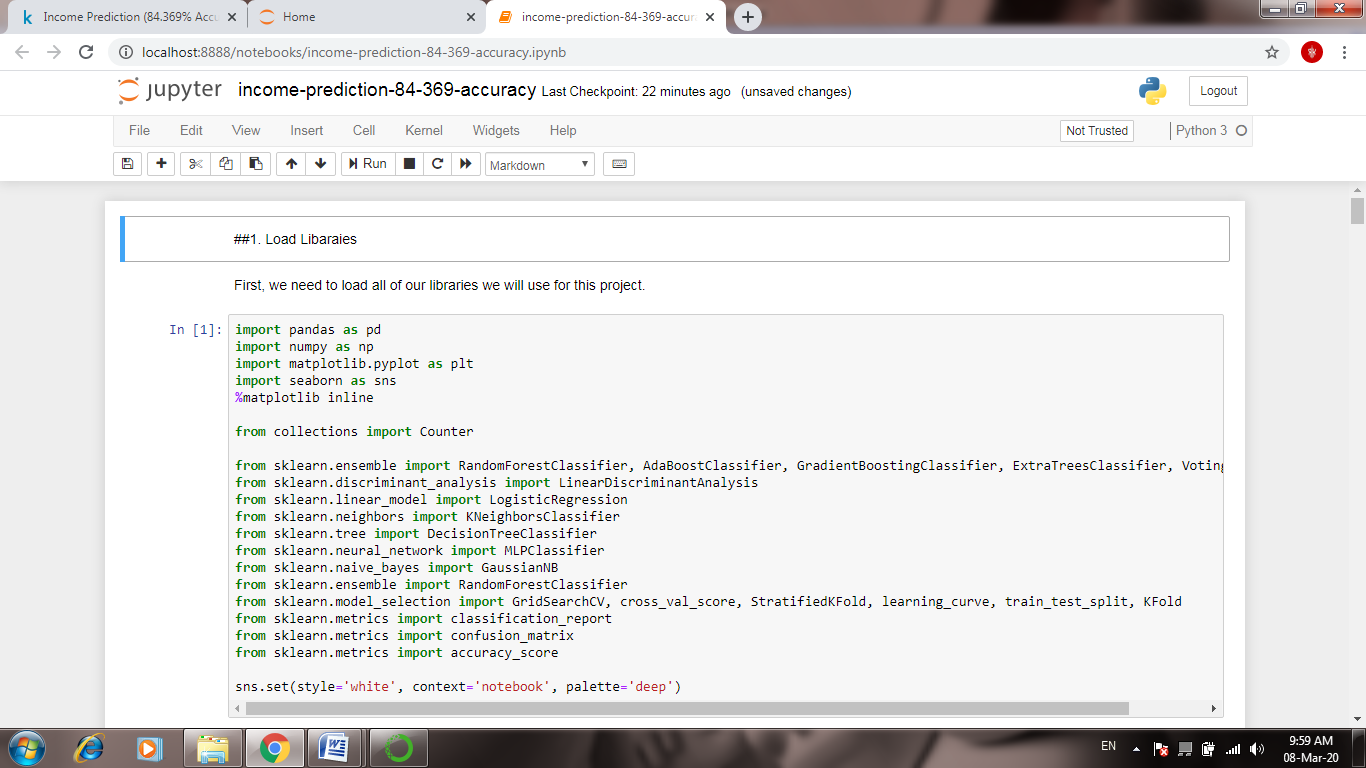
**COLUMNS:**

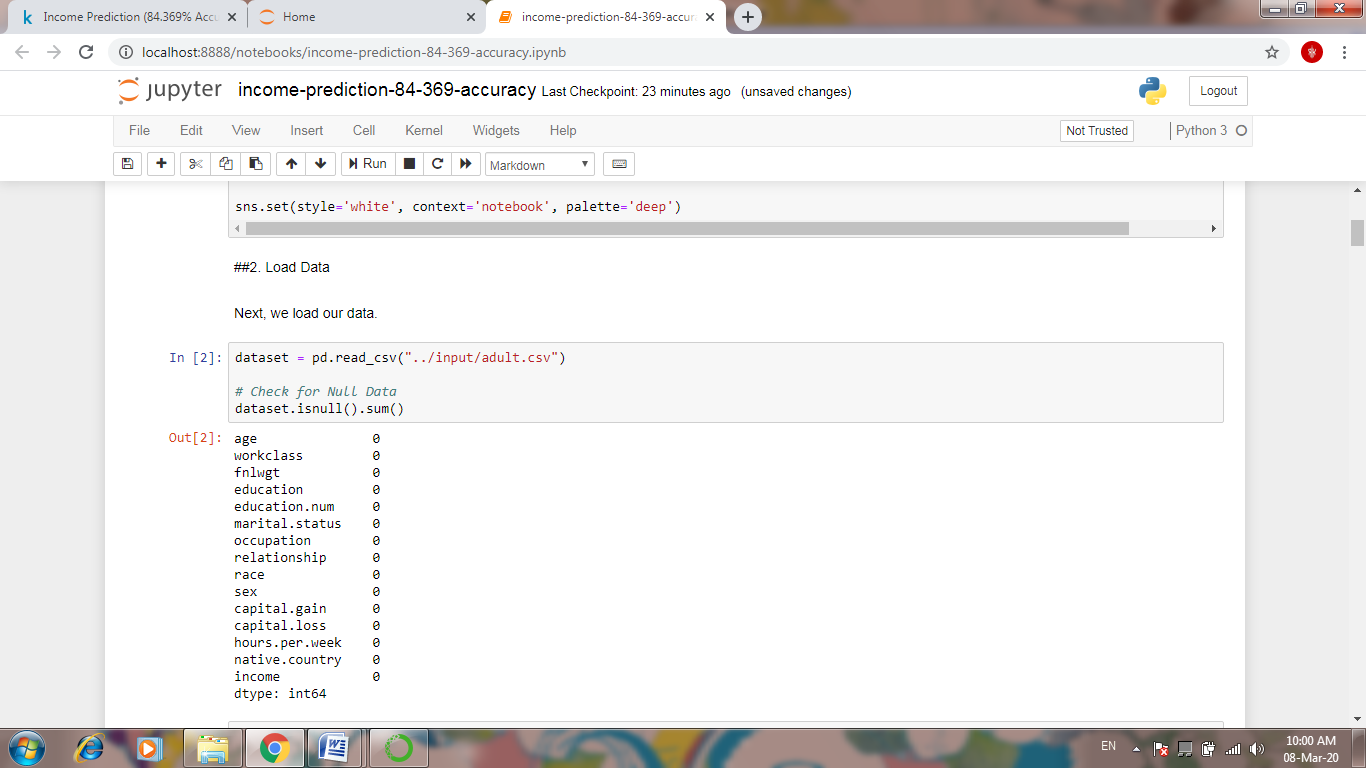
* age
* workclass
* fnlwgt
* education
* education.num
* marital.status
* occupation
* relationship
* race
* sex
* capital.gain
* capital.loss
* hours.per.week
* native.country
* income

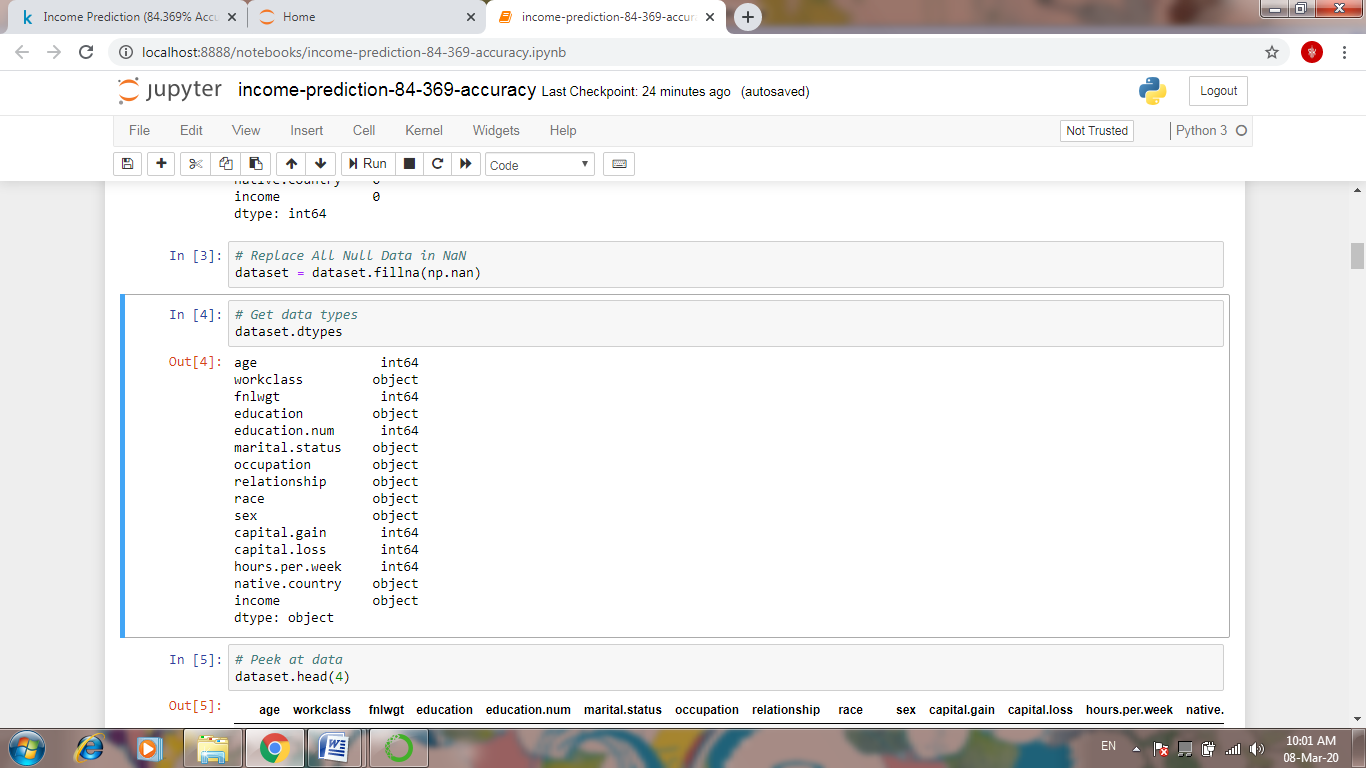
**PRE\_PROCESSING:**

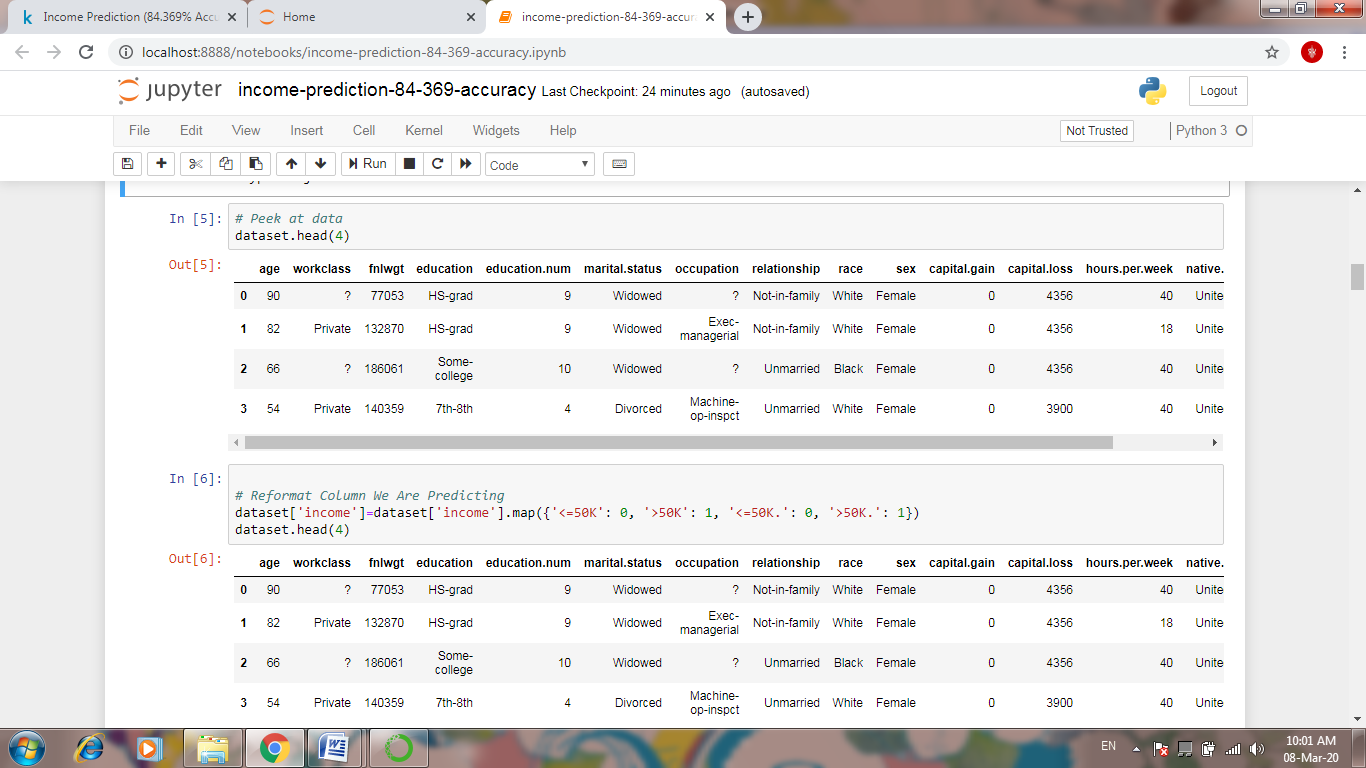
Pre-processing refers to the transformations applied to our data before feeding it to the algorithm.

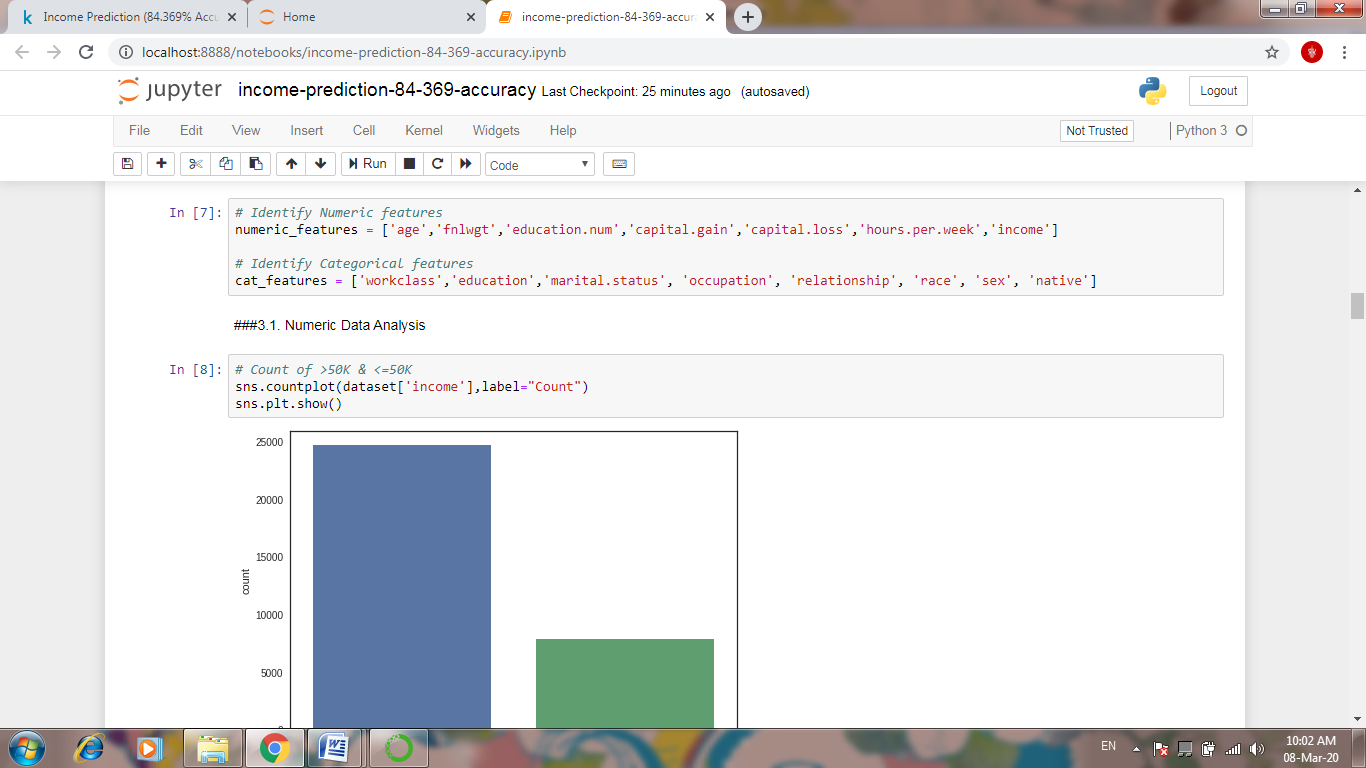
1. Load Libraries
2. Load Data
3. Analyze Data
4. Feature Engineering
5. Modeling
6. Algorithm Tuning
7. Finalizing the Model

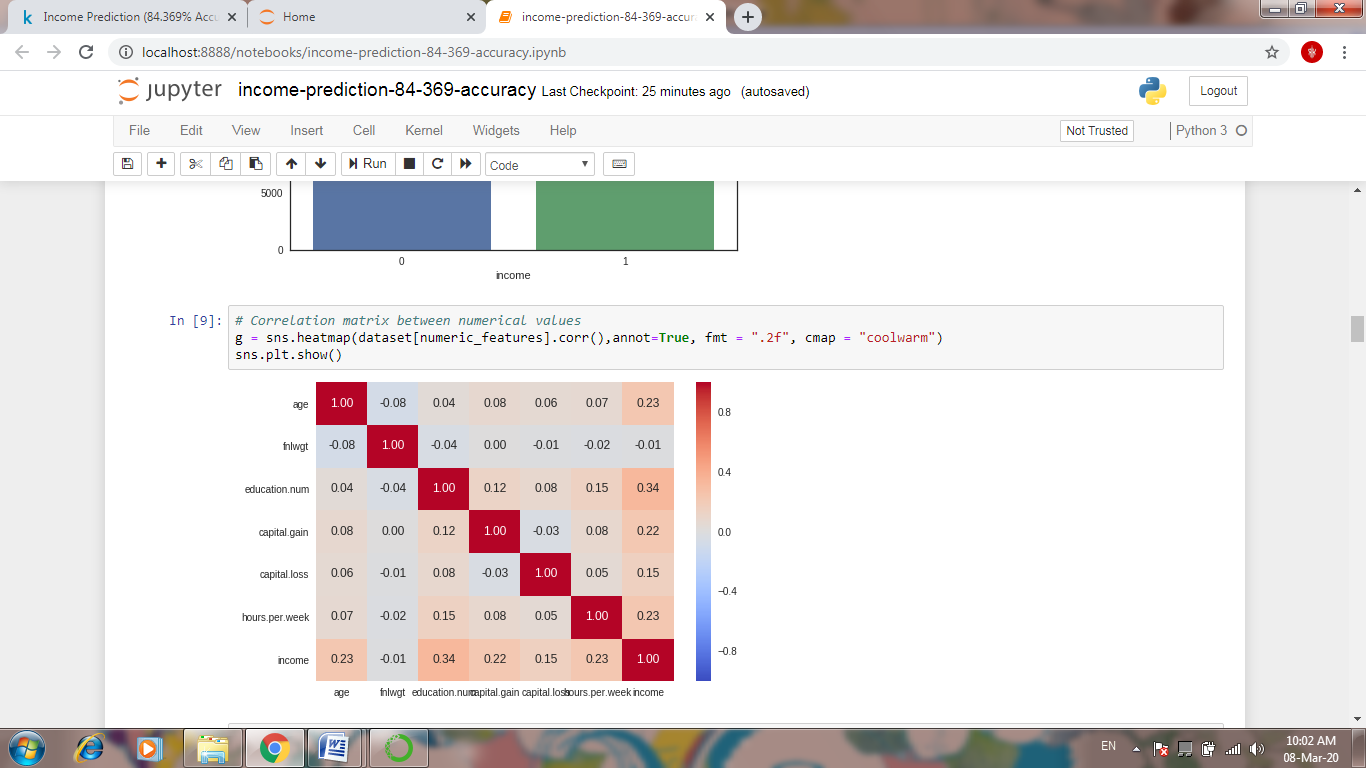


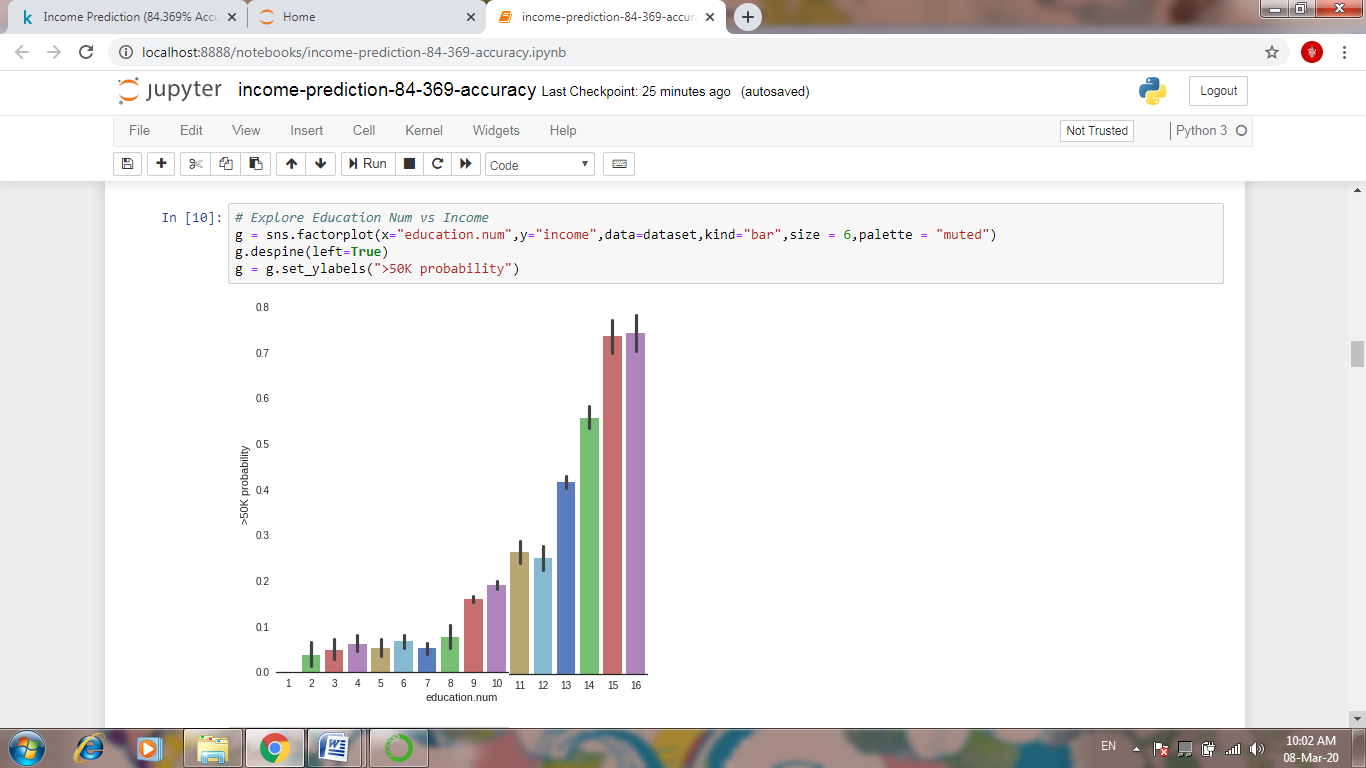


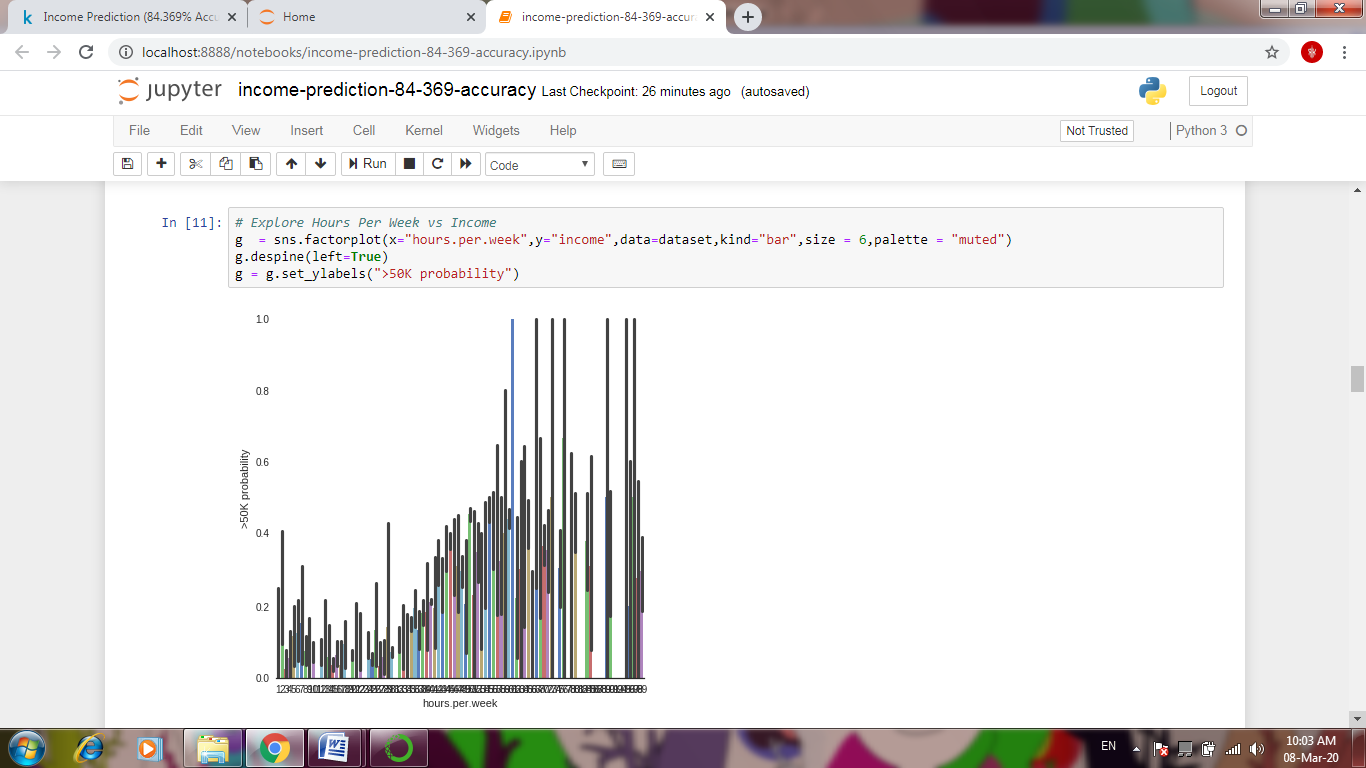


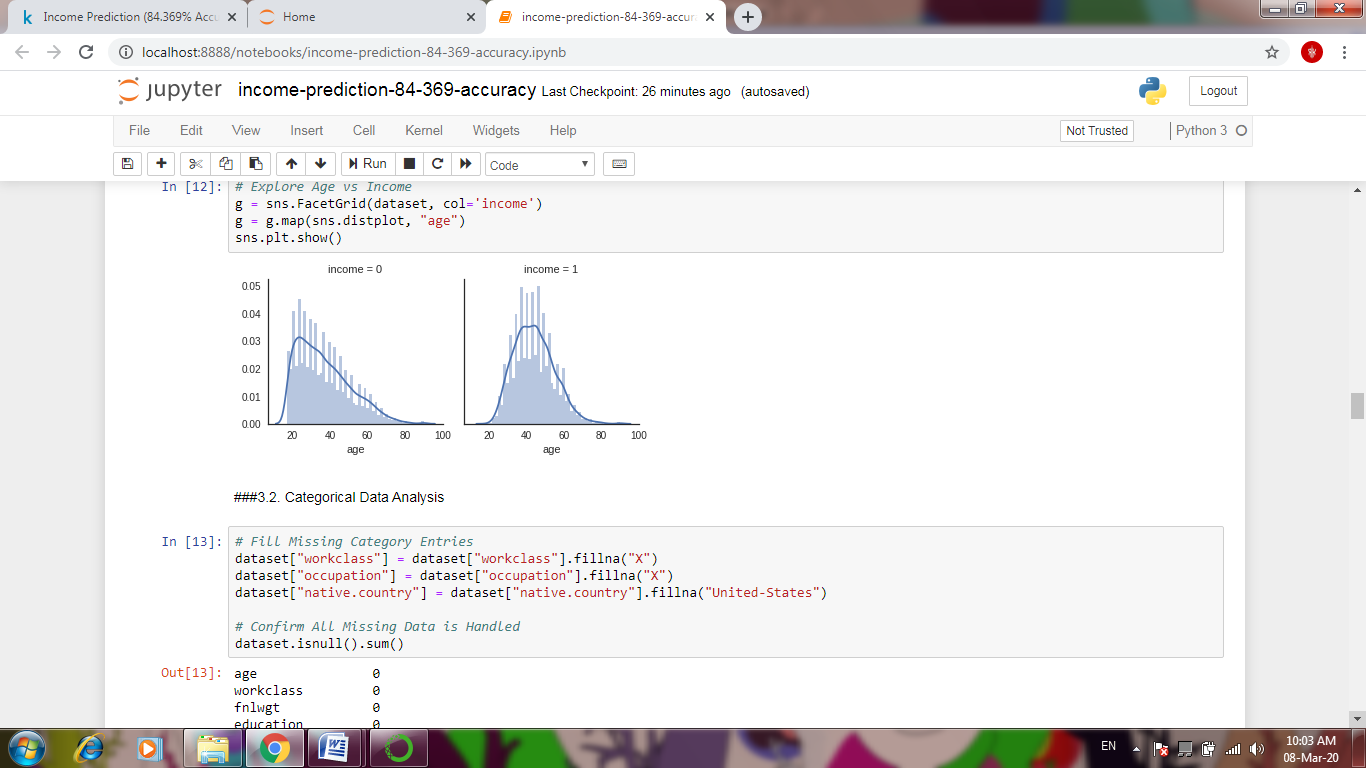


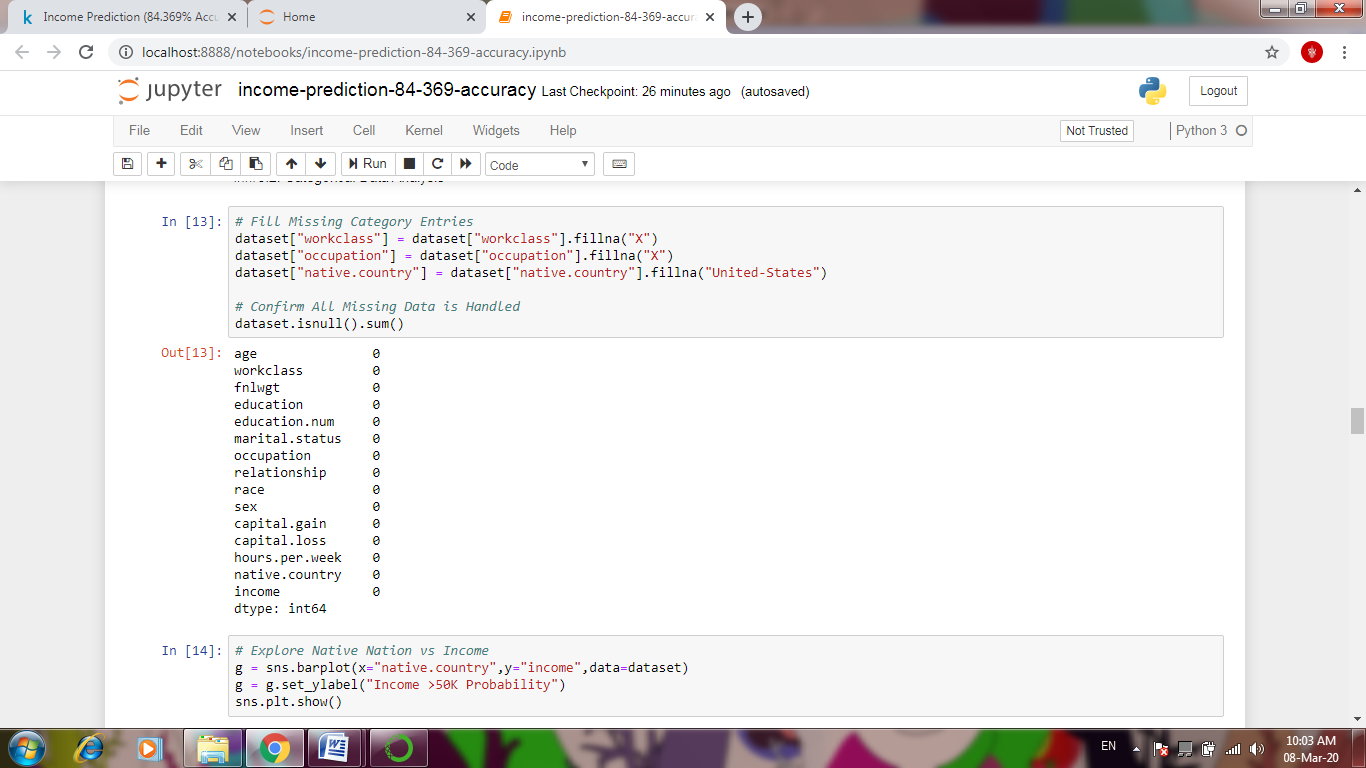


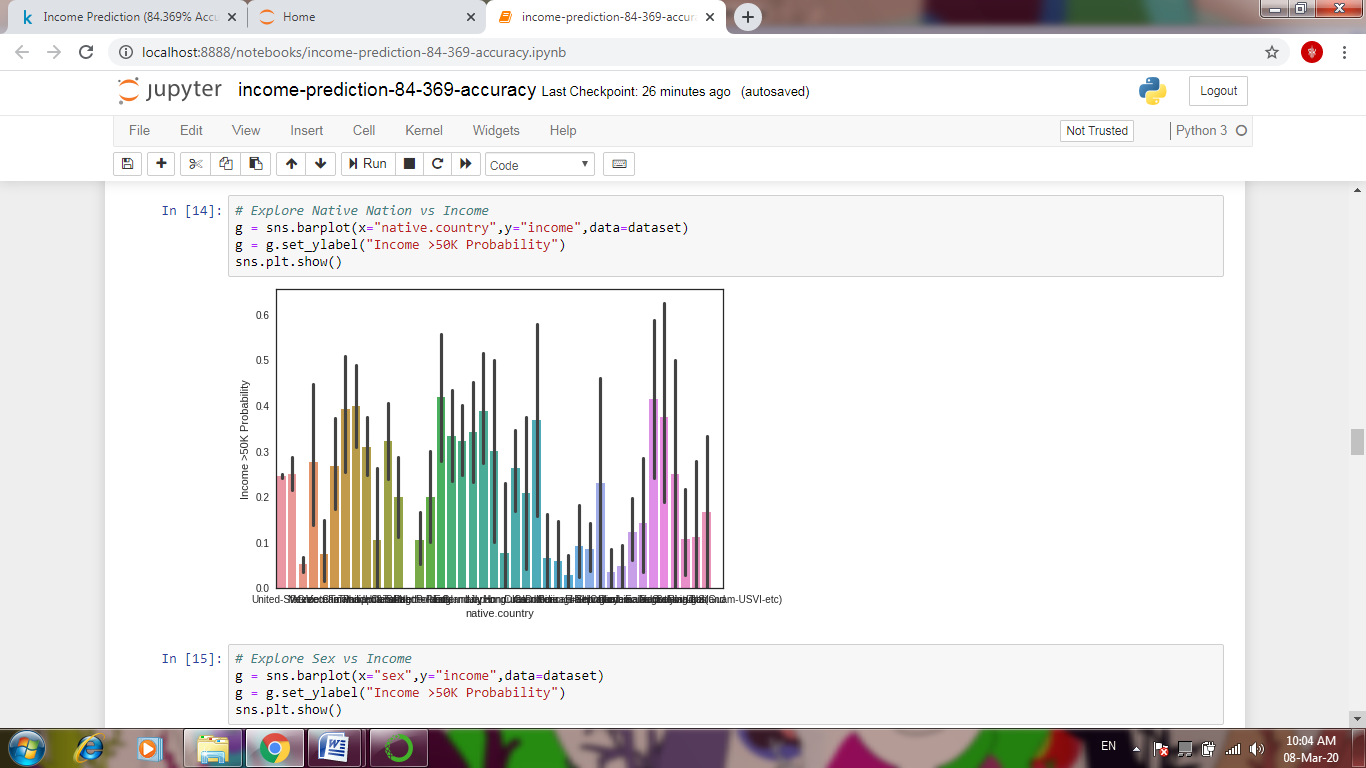


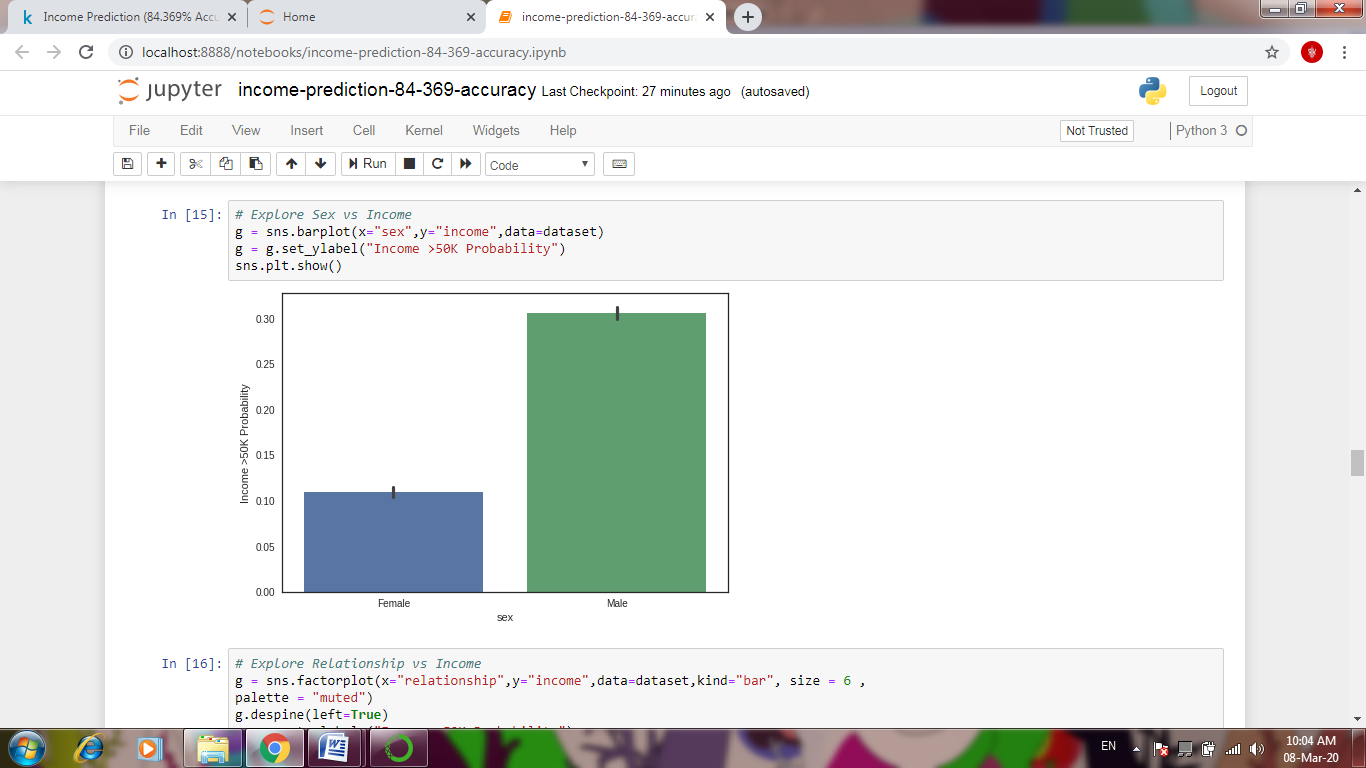


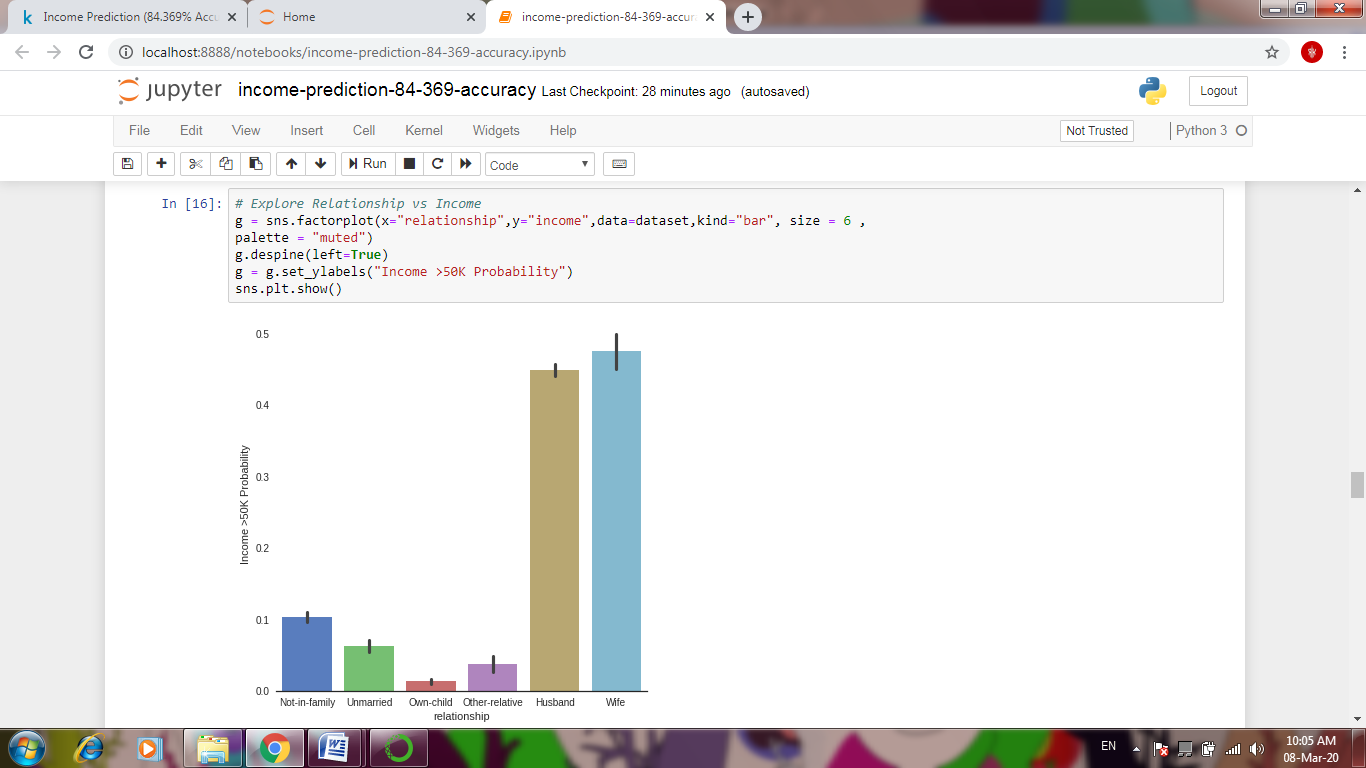


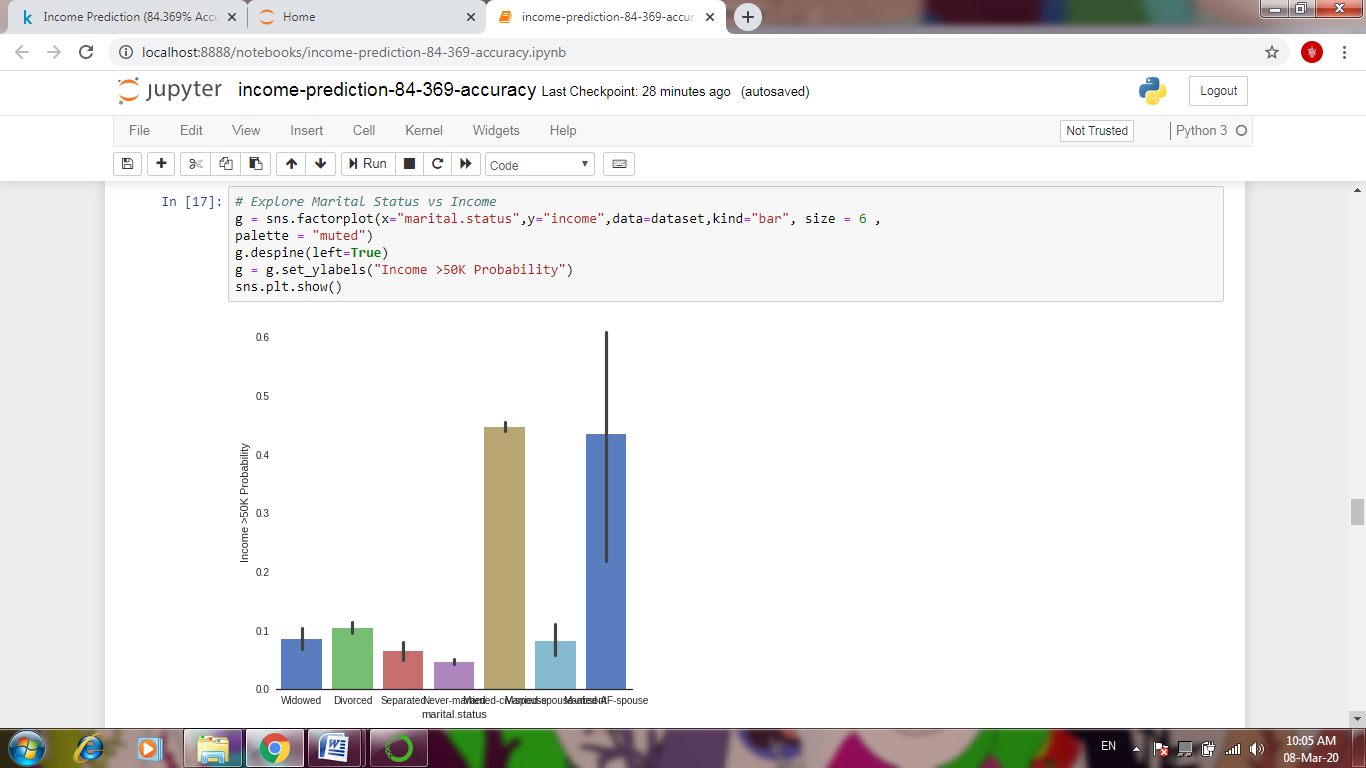


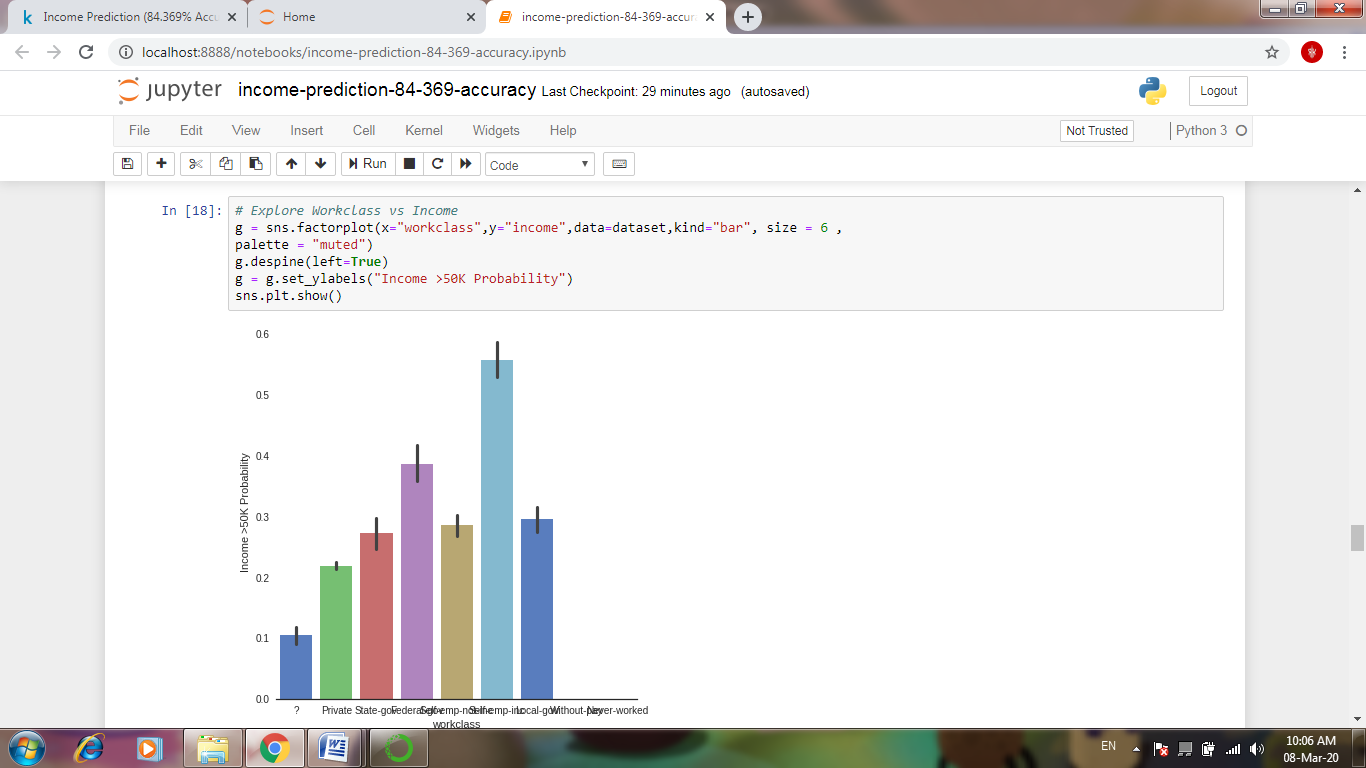


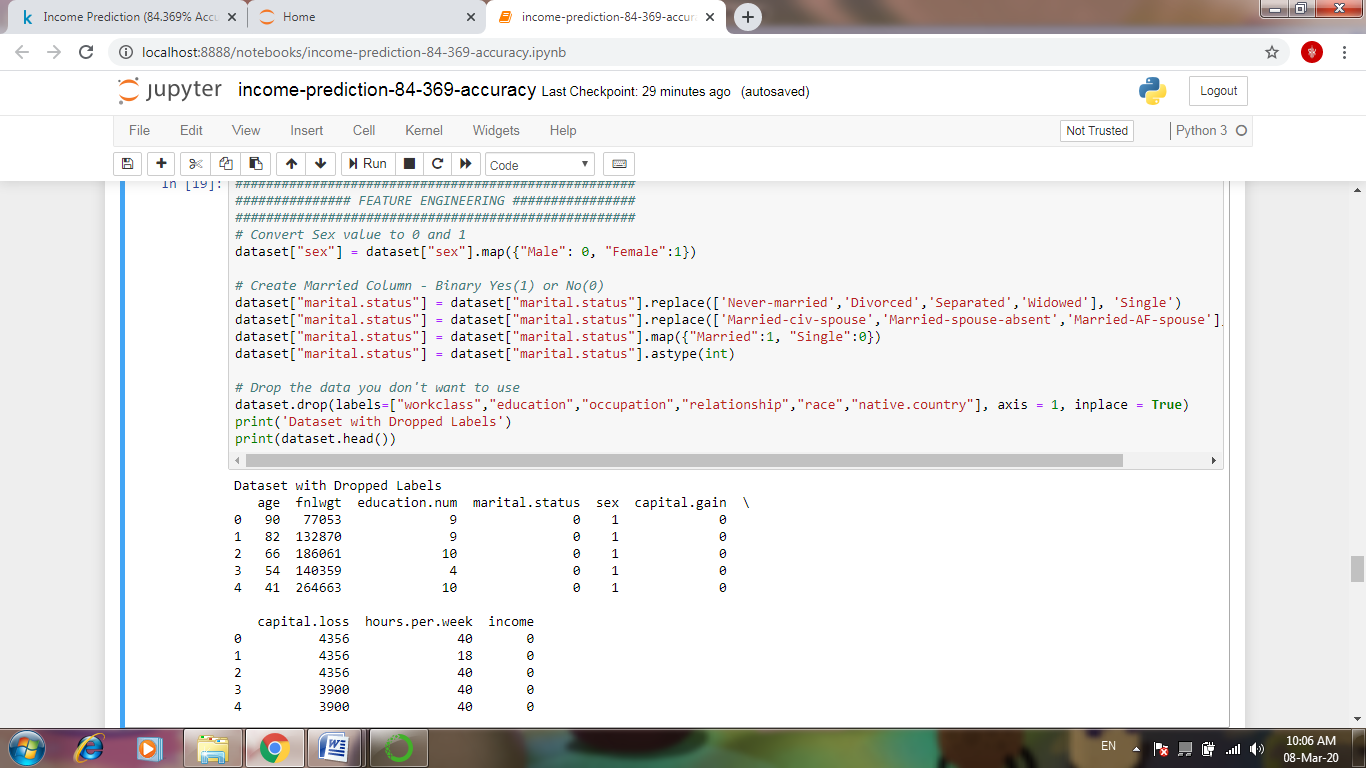


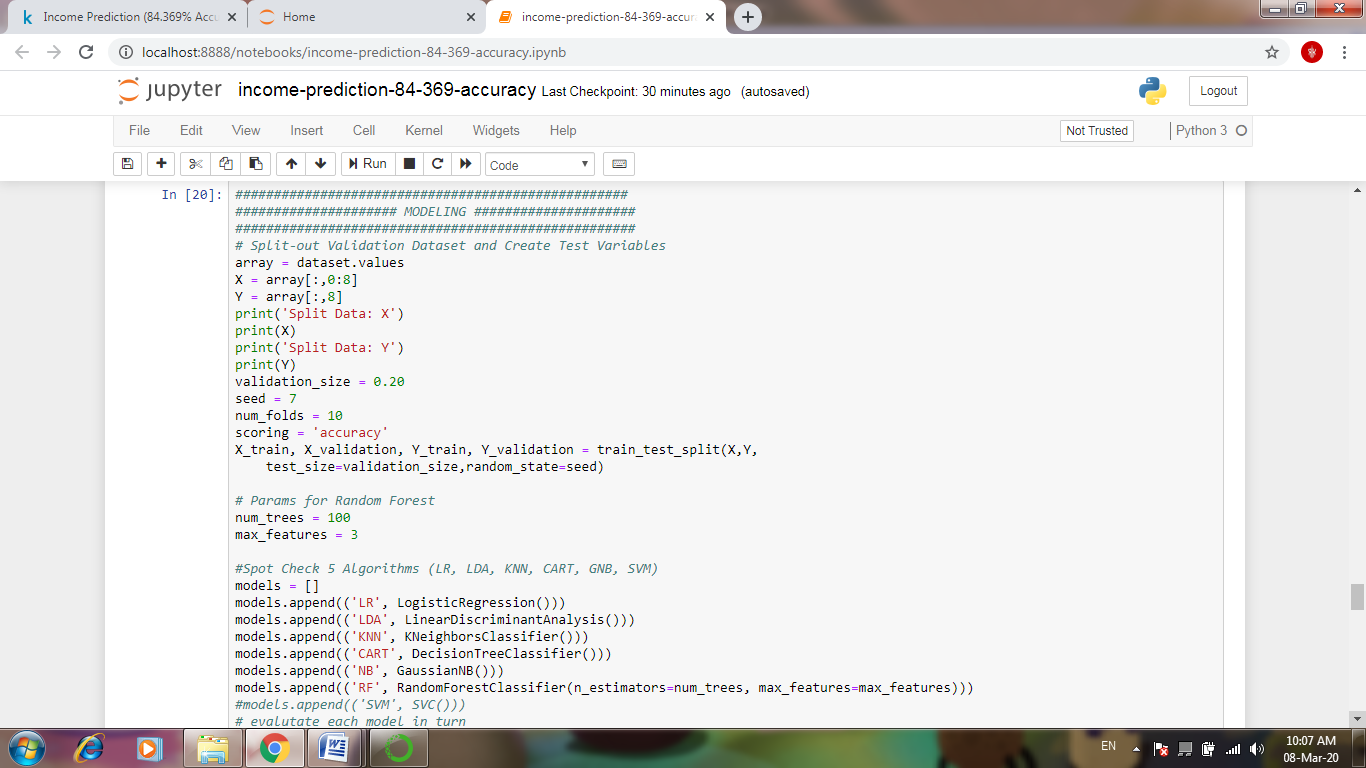


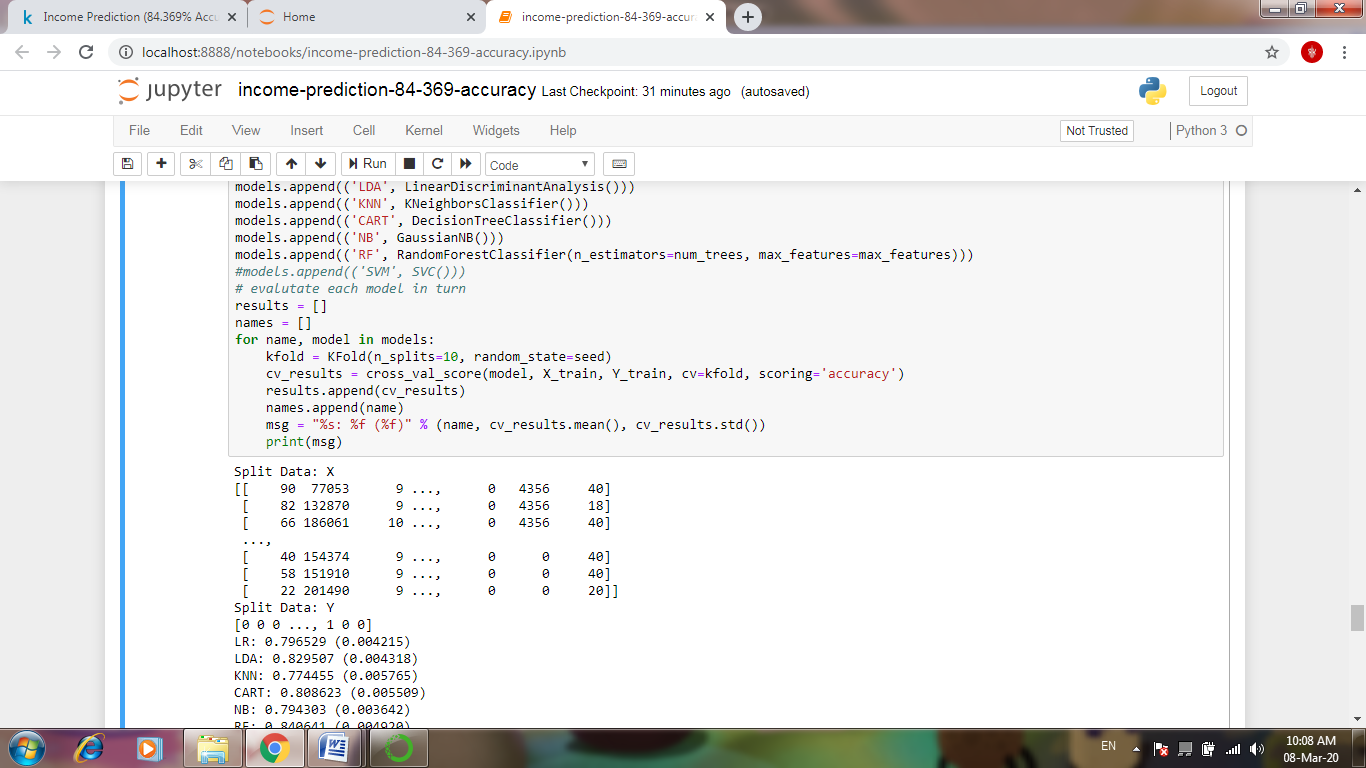


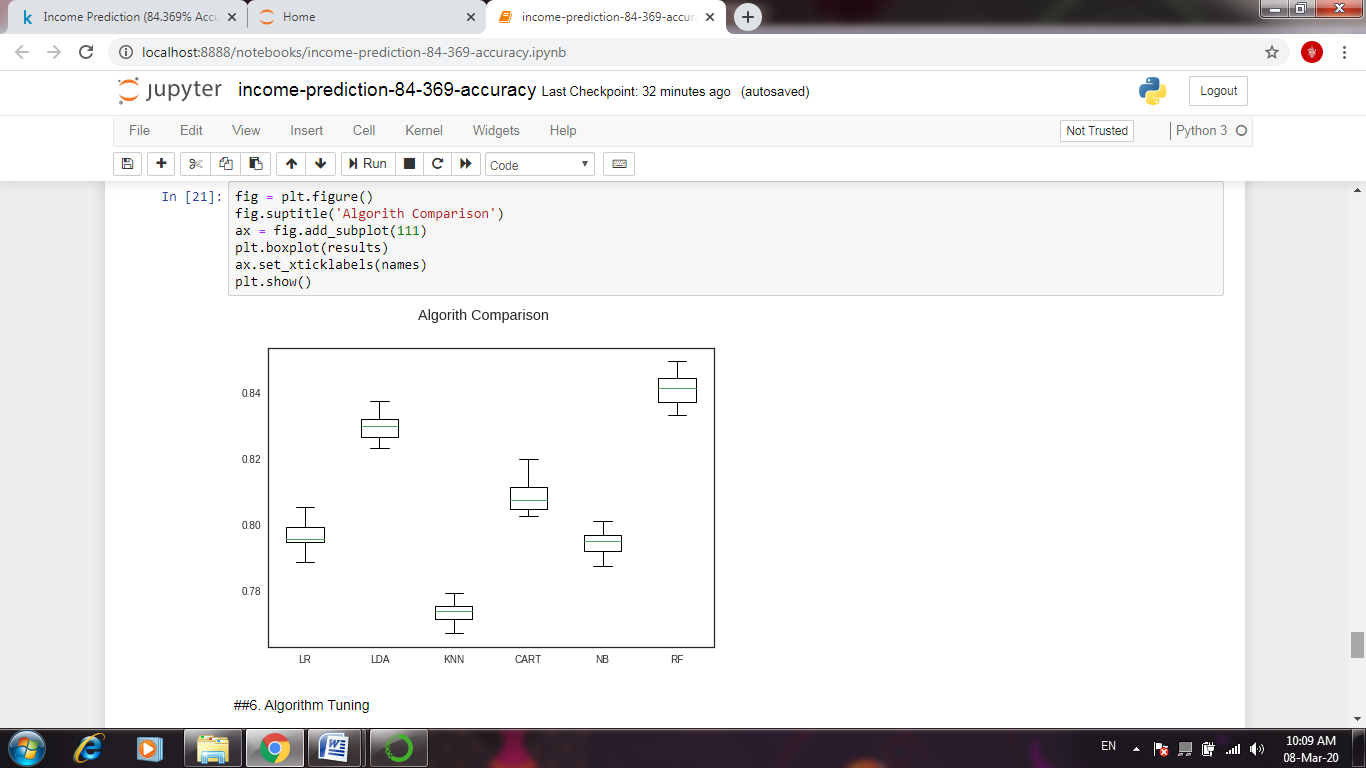


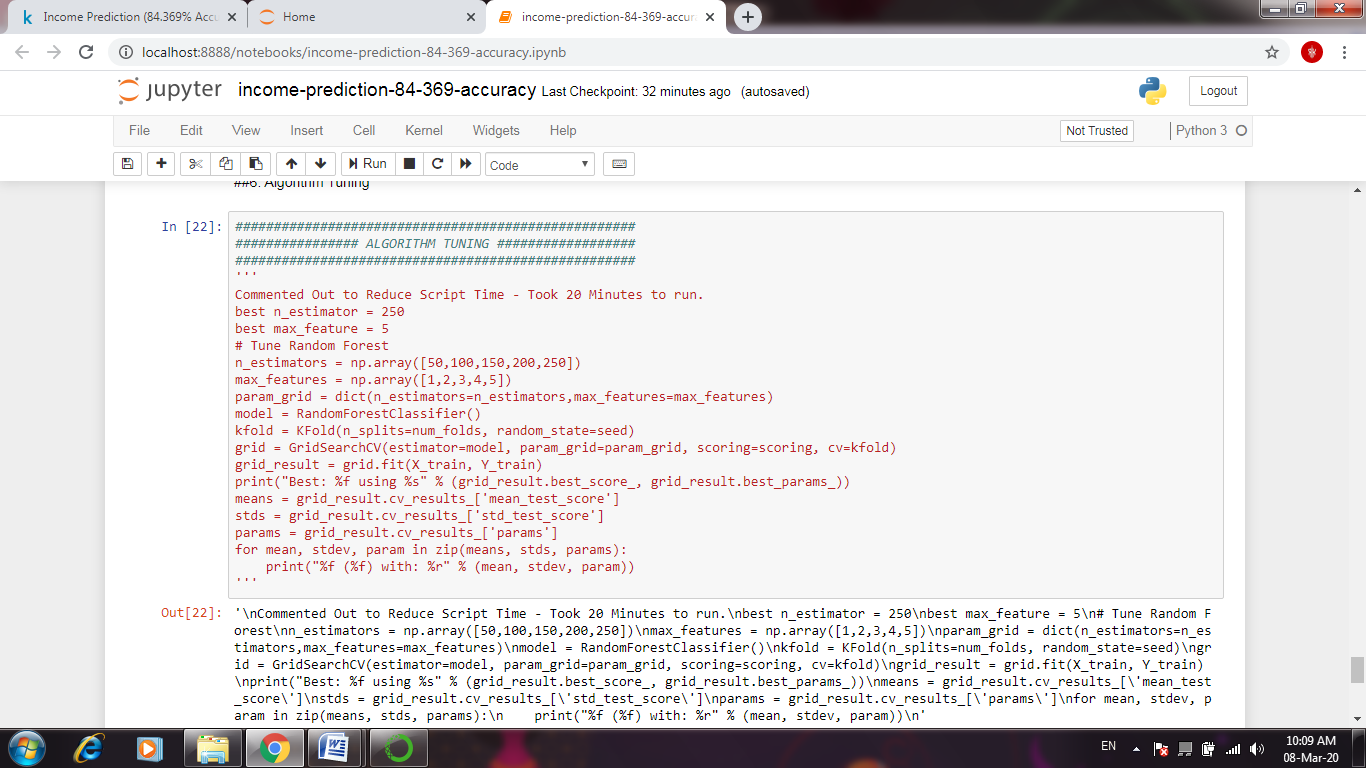


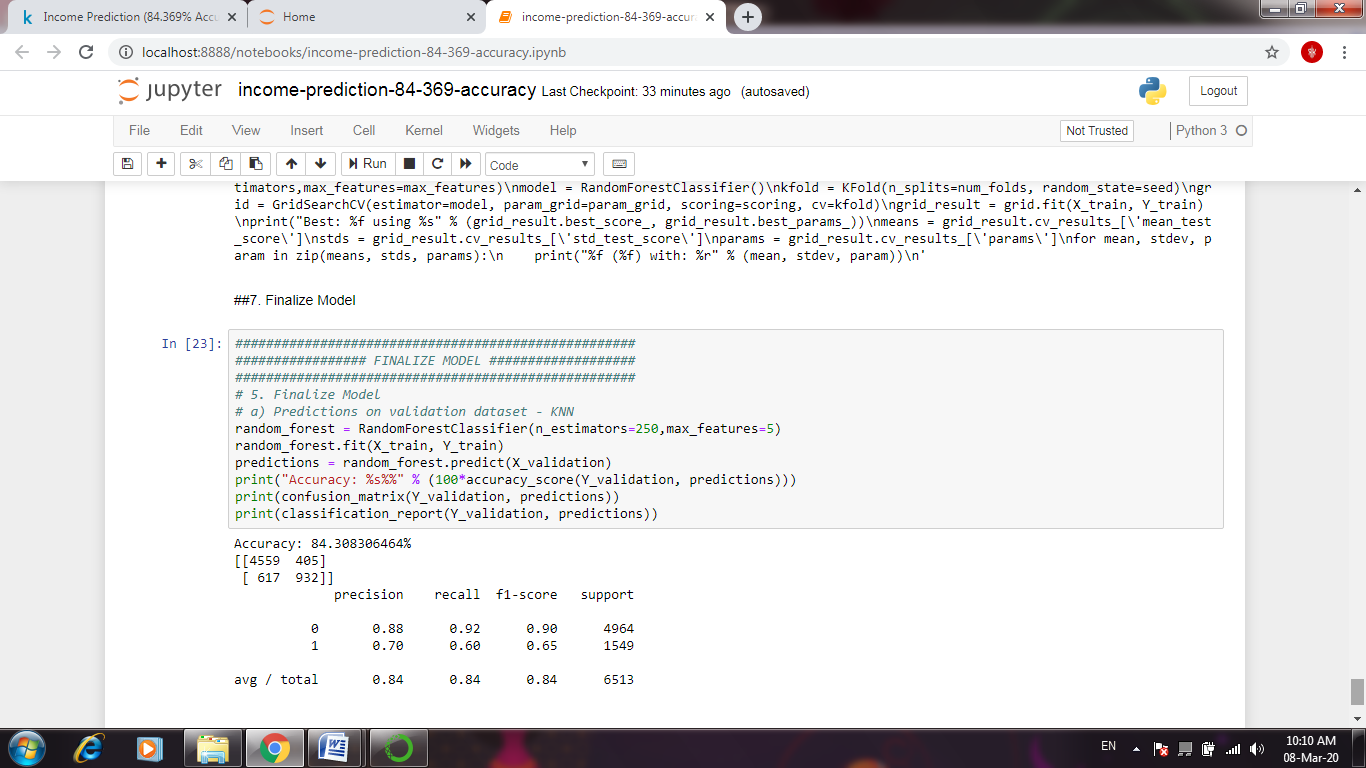












CODING:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

from collections import Counter

from sklearn.ensemble import RandomForestClassifier,

AdaBoostClassifier, GradientBoostingClassifier, ExtraTreesClassifier,

VotingClassifier

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.neural\_network import MLPClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import GridSearchCV, cross\_val\_score,

StratifiedKFold, learning\_curve, train\_test\_split, KFold

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import accuracy\_score

sns.set(style='white', context='notebook', palette='deep')

**2. Load Data**

dataset = pd.read\_csv("../input/adult.csv")

dataset = dataset.fillna(np.nan)

dataset.dtypes

dataset.head(4)

dataset['income']=dataset['income'].map({'<=50K': 0, '>50K': 1, '<=50K.':

0, '>50K.': 1})

dataset.head(4)

**3. Analyze Data**

numeric\_features =

['age','fnlwgt','education.num','capital.gain','capital.loss','hours.per.week','i

ncome']

cat\_features = ['workclass','education','marital.status', 'occupation',

'relationship', 'race', 'sex', 'native']

**4. Feature Engineering**

# Convert Sex value to 0 and 1

dataset["sex"] = dataset["sex"].map({"Male": 0, "Female":1})

# Create Married Column - Binary Yes(1) or No(0)

dataset["marital.status"] = dataset["marital.status"].replace(['Never-

married','Divorced','Separated','Widowed'], 'Single')

dataset["marital.status"] = dataset["marital.status"].replace(['Married-civ-

spouse','Married-spouse-absent','Married-AF-spouse'], 'Married')

dataset["marital.status"] = dataset["marital.status"].map({"Married":1,

"Single":0})

dataset["marital.status"] = dataset["marital.status"].astype(int)

# Drop the data you don't want to use

dataset.drop(labels=["workclass","education","occupation","relationship",

"race","native.country"], axis = 1, inplace = True)

print('Dataset with Dropped Labels')

print(dataset.head())

**5. Modeling**

# Split-out Validation Dataset and Create Test Variables

array = dataset.values

X = array[:,0:8]

Y = array[:,8]

print('Split Data: X')

print(X)

print('Split Data: Y')

print(Y)

validation\_size = 0.20

seed = 7

num\_folds = 10

scoring = 'accuracy'

X\_train, X\_validation, Y\_train, Y\_validation = train\_test\_split(X,Y,

test\_size=validation\_size,random\_state=seed)

# Params for Random Forest

num\_trees = 100

max\_features = 3

#Spot Check 5 Algorithms (LR, LDA, KNN, CART, GNB, SVM)

models = []

models.append(('LR', LogisticRegression()))

models.append(('LDA', LinearDiscriminantAnalysis()))

models.append(('KNN', KNeighborsClassifier()))

models.append(('CART', DecisionTreeClassifier()))

models.append(('NB', GaussianNB()))

models.append(('RF', RandomForestClassifier(n\_estimators=num\_trees,

max\_features=max\_features)))

#models.append(('SVM', SVC()))

# evalutate each model in turn

results = []

names = []

for name, model in models:

kfold = KFold(n\_splits=10, random\_state=seed)

cv\_results = cross\_val\_score(model, X\_train, Y\_train, cv=kfold,

scoring='accuracy')

results.append(cv\_results)

names.append(name)

msg = "%s: %f (%f)" % (name, cv\_results.mean(), cv\_results.std())

print(msg)

**6. Algorithm Tuning**

Commented Out to Reduce Script Time - Took 20 Minutes to run.

best n\_estimator = 250

best max\_feature = 5

# Tune Random Forest

n\_estimators = np.array([50,100,150,200,250])

max\_features = np.array([1,2,3,4,5])

param\_grid = dict(n\_estimators=n\_estimators,max\_features=max\_features)

model = RandomForestClassifier()

kfold = KFold(n\_splits=num\_folds, random\_state=seed)

grid = GridSearchCV(estimator=model, param\_grid=param\_grid, scoring=scoring, cv=kfold)

grid\_result = grid.fit(X\_train, Y\_train)

print("Best: %f using %s" % (grid\_result.best\_score\_, grid\_result.best\_params\_))

means = grid\_result.cv\_results\_['mean\_test\_score']

stds = grid\_result.cv\_results\_['std\_test\_score']

params = grid\_result.cv\_results\_['params']

for mean, stdev, param in zip(means, stds, params):

print("%f (%f) with: %r" % (mean, stdev, param))

1. **Finalize Model**

Predictions on validation dataset - KNN

random\_forest = RandomForestClassifier(n\_estimators=250,max\_features=5)

random\_forest.fit(X\_train, Y\_train)

predictions = random\_forest.predict(X\_validation)

print("Accuracy: %s%%" % (100\*accuracy\_score(Y\_validation, predictions)))

print(confusion\_matrix(Y\_validation, predictions))

print(classification\_report(Y\_validation, predictions))

**CONCLUSION:**

In this case-study we constructed a model that is able to accurately predict whether an individual earns over $50k/yr based on various data like marital status, education, age or nationality. The model might be tuned to meet our business needs which could be to classify high-income customers more accurately.

**URL:**