**Problem Statement 1: - Mandatory**

Explore the given data set with EDA techniques and build a suitable model for predicting whether the salary of the person is >50k or not and visualize the results. NOTE: the algorithm used for the model must be built from the scratch.

**PROBLEM STATEMENT AND ANALYSIS:**

[Logistic regression](https://www.statisticssolutions.com/academic-solutions/membership-resources/member-profile/data-analysis-plan-templates/data-analysis-plan-logistic-regression/) is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary).  Like all regression analyses, the logistic regression is a predictive analysis.  Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

A support vector machine (SVM) is machine learning algorithm that analyzes data for classification and regression analysis. SVM is a supervised learning method that looks at data and sorts it into one of two categories. An SVM outputs a map of the sorted data with the margins between the two as far apart as possible. SVMs are used in text categorization, image classification, handwriting recognition and in the sciences.A support vector machine is also known as a support vector network (SVN).

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

**CODE:**

#DATA PREPROCESSING

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn import preprocessing

dt = pd.read\_csv('train.csv')

dt.head()

dt.info()

dt.shape

dt.describe()

dt.isnull().sum()

train=dt.dropna()

train.isnull().sum()

sns.heatmap(train.corr(),cmap='coolwarm')

#encoding

train.columns = ['age', 'workclass', 'fnlwgt', 'education', 'education.num','marital.status','occupation','relationship','race','sex','captial.gain','capital.loss','hours.per.week','native.country','target']

from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()

train['sex']=le.fit\_transform(train['sex'])

train['marital.status']=le.fit\_transform(train['marital.status'])

train['education']=le.fit\_transform(train['education'])

train['relationship']=le.fit\_transform(train['relationship'])

train['race']=le.fit\_transform(train['race'])

train['occupation']=le.fit\_transform(train['occupation'])

train['workclass']=le.fit\_transform(train['workclass'])

train['native.country']=le.fit\_transform(train['native.country'])

train.head()

dd= pd.read\_csv('test.csv')

dd.head()

dd.shape

dd.info()

dd.describe()

dd.isnull().sum()

test=dt.dropna()

test.isnull().sum()

#encoding

test.columns = ['age', 'workclass', 'fnlwgt', 'education', 'education.num','marital.status','occupation','relationship','race','sex','captial.gain','capital.loss','hours.per.week','native.country','target']

from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()

test['sex']=le.fit\_transform(test['sex'])

test['marital.status']=le.fit\_transform(test['marital.status'])

test['education']=le.fit\_transform(test['education'])

test['relationship']=le.fit\_transform(test['relationship'])

test['race']=le.fit\_transform(test['race'])

test['occupation']=le.fit\_transform(test['occupation'])

test['workclass']=le.fit\_transform(test['workclass'])

test['native.country']=le.fit\_transform(test['native.country'])

test.head()

from sklearn.model\_selection import train\_test\_split

X = train[['age', 'workclass', 'fnlwgt', 'education', 'education.num','marital.status','occupation','relationship','race','sex','captial.gain','capital.loss','hours.per.week','native.country']]

y = train['target']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33)

#logestic regression

from sklearn.linear\_model import LogisticRegression

logmodel = LogisticRegression()

logmodel.fit(X\_train,y\_train)

valid\_predictions = logmodel.predict(X\_test)

from sklearn.metrics import classification\_report

print(classification\_report(y\_test,valid\_predictions))

#svm

from sklearn.svm import SVC

svc\_model = SVC()

svc\_model.fit(X\_train,y\_train)

svm\_predictions = svc\_model.predict(X\_test)

from sklearn.metrics import classification\_report,confusion\_matrix

print(confusion\_matrix(y\_test,svm\_predictions))

print(classification\_report(y\_test,svm\_predictions))

#random forest

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n\_estimators=600)

rfc.fit(X\_train,y\_train)

rf\_predictions = rfc.predict(X\_test)

print(classification\_report(y\_test,rf\_predictions))

#testing

rfc = RandomForestClassifier(n\_estimators=600)

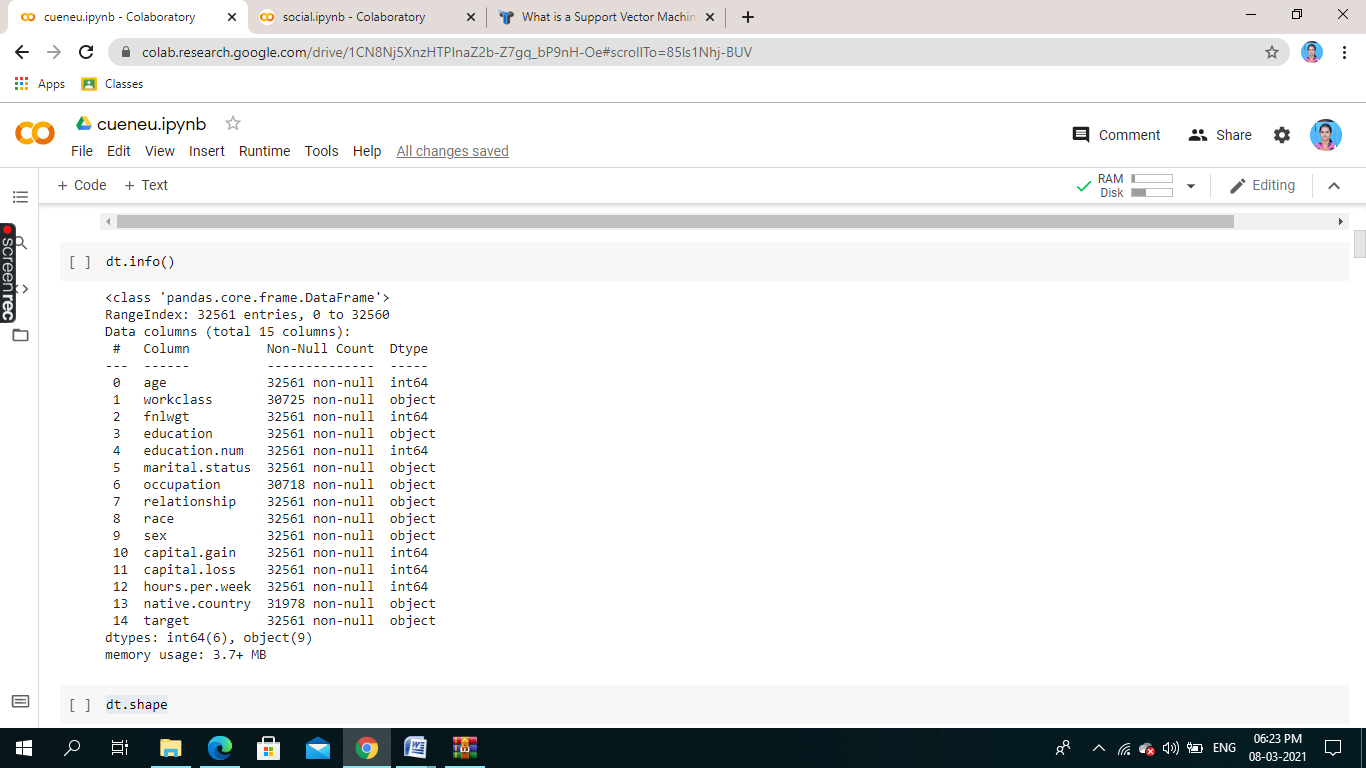
rfc.fit(X,y)

final\_predictions = rfc.predict(test.drop(['target'],axis=1))

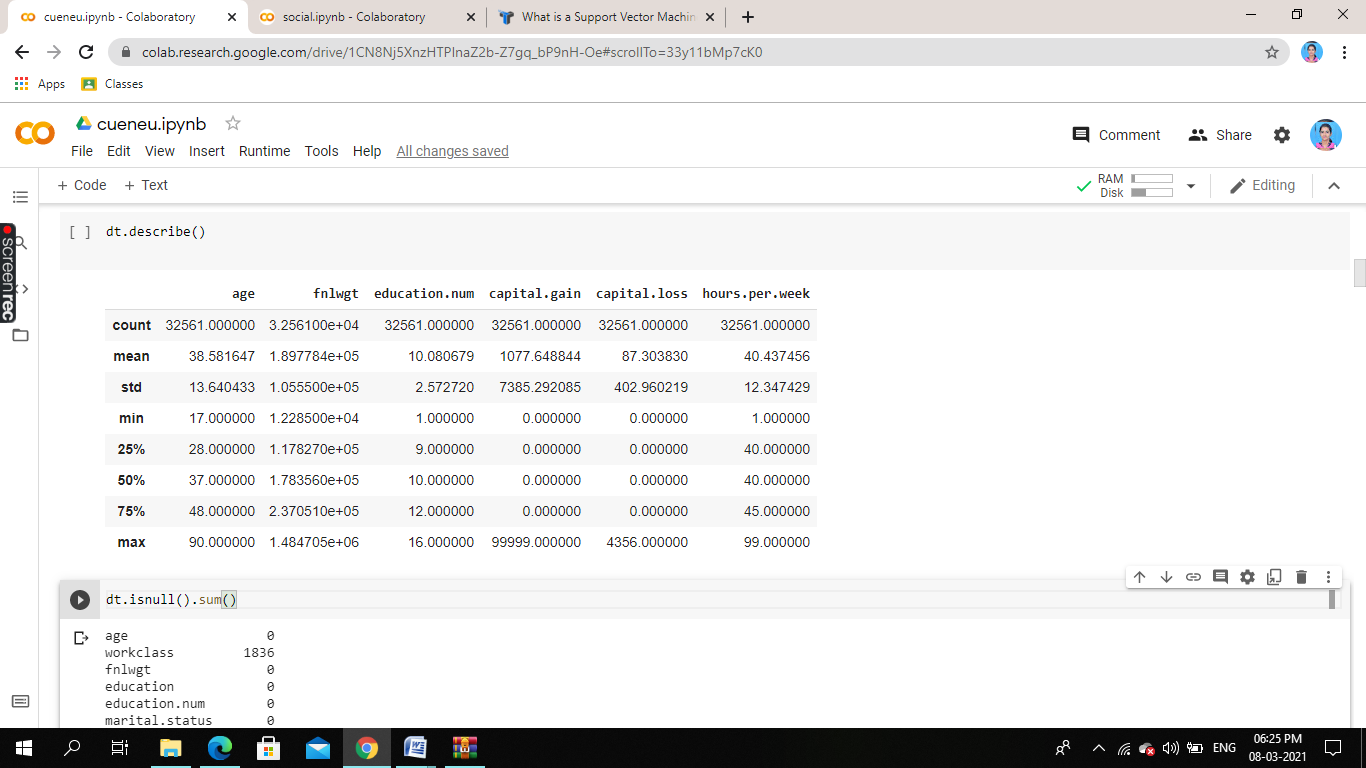
print(confusion\_matrix(test['target'], final\_predictions))

**OUTPUT:**

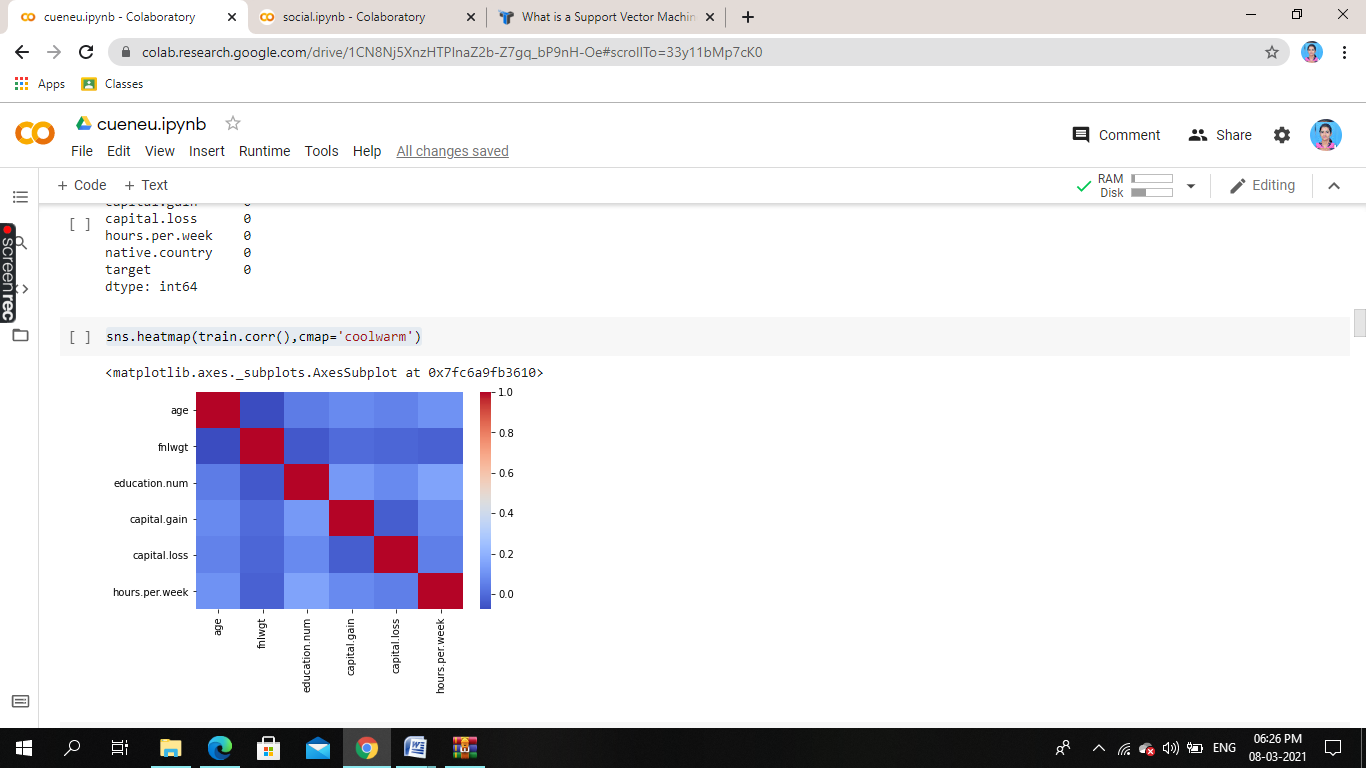
**INFO**

****

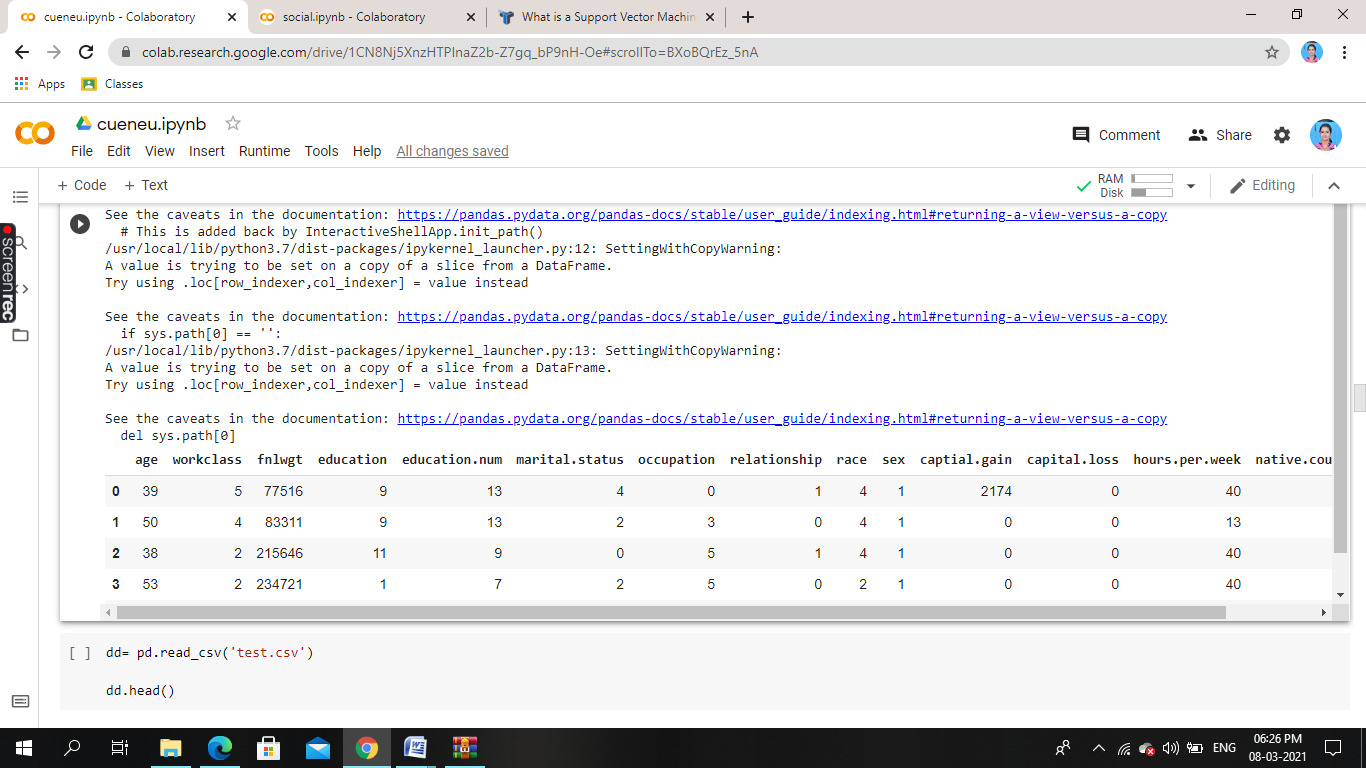
**DESCRIBE**

****

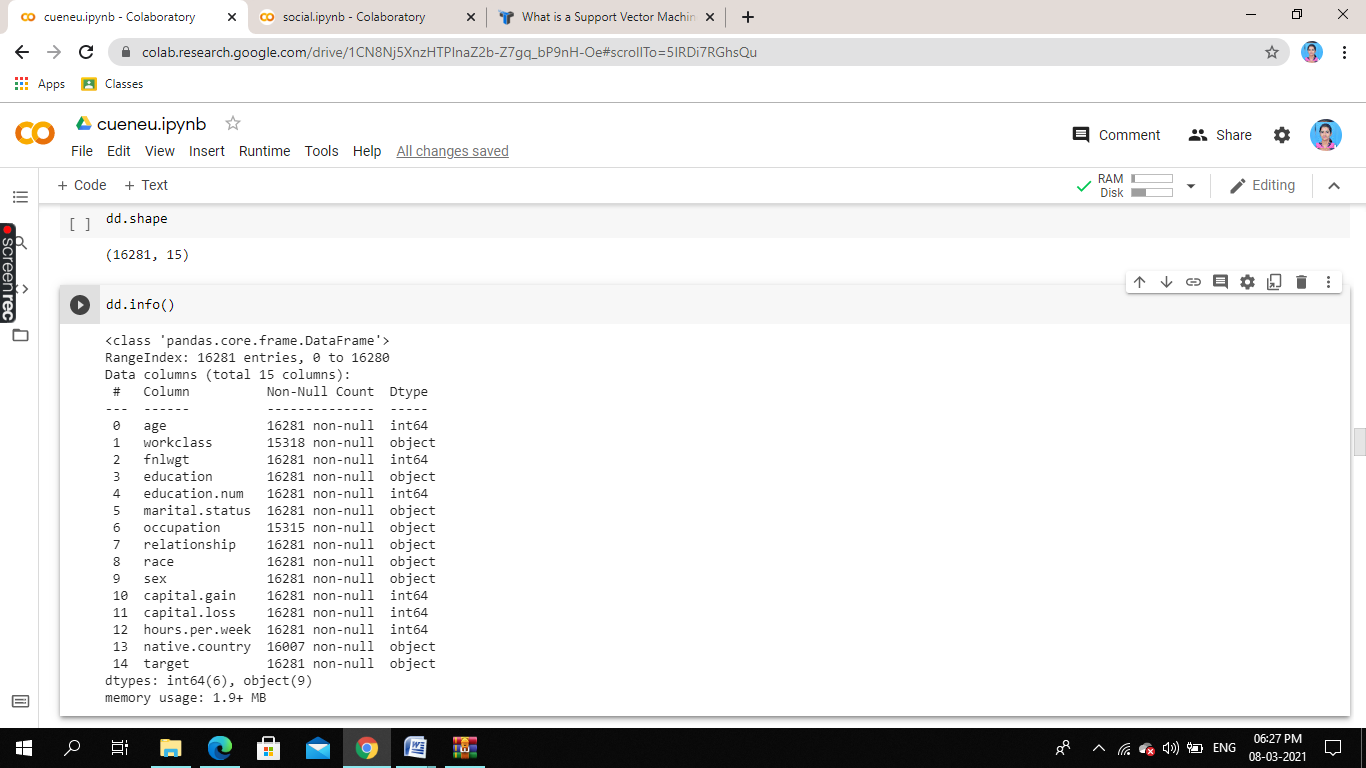
**HEATMAP**

****

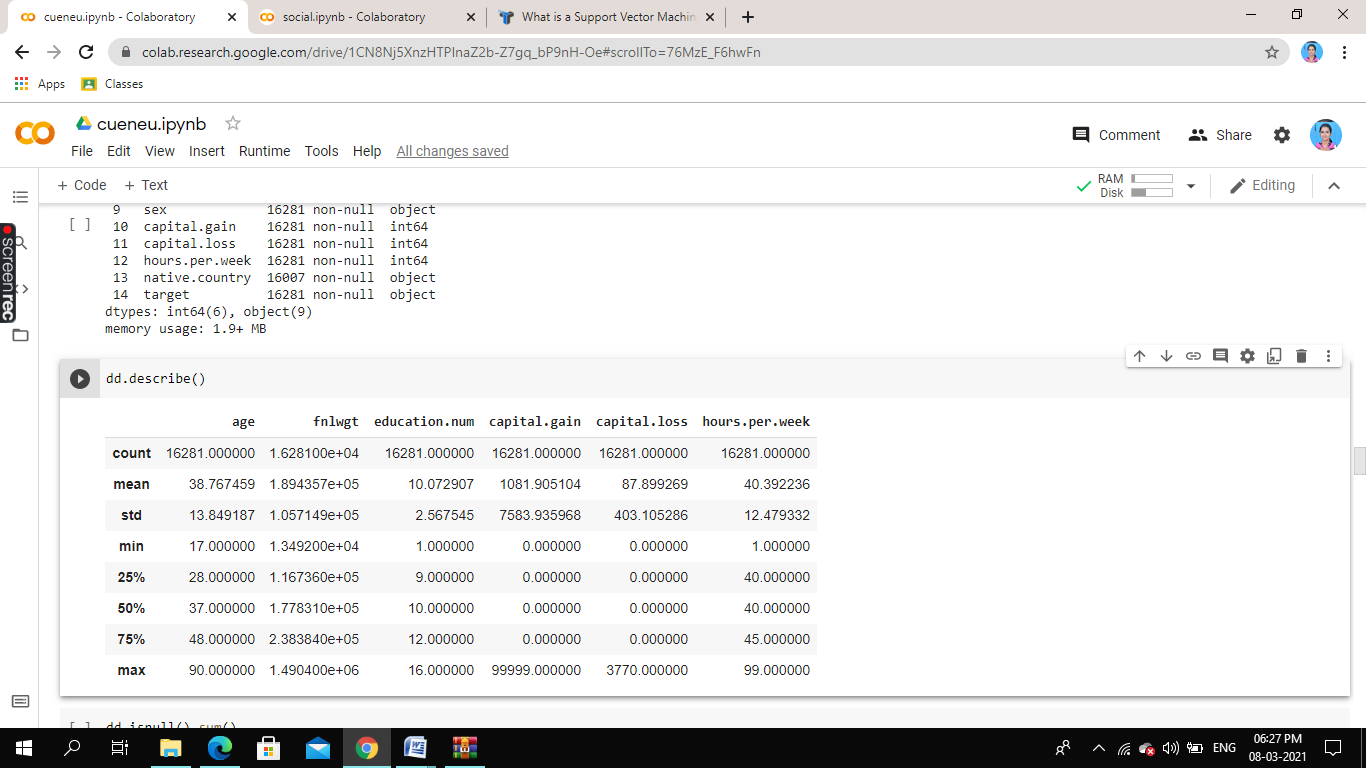
**ENCODED TRAIN VALUE**

****

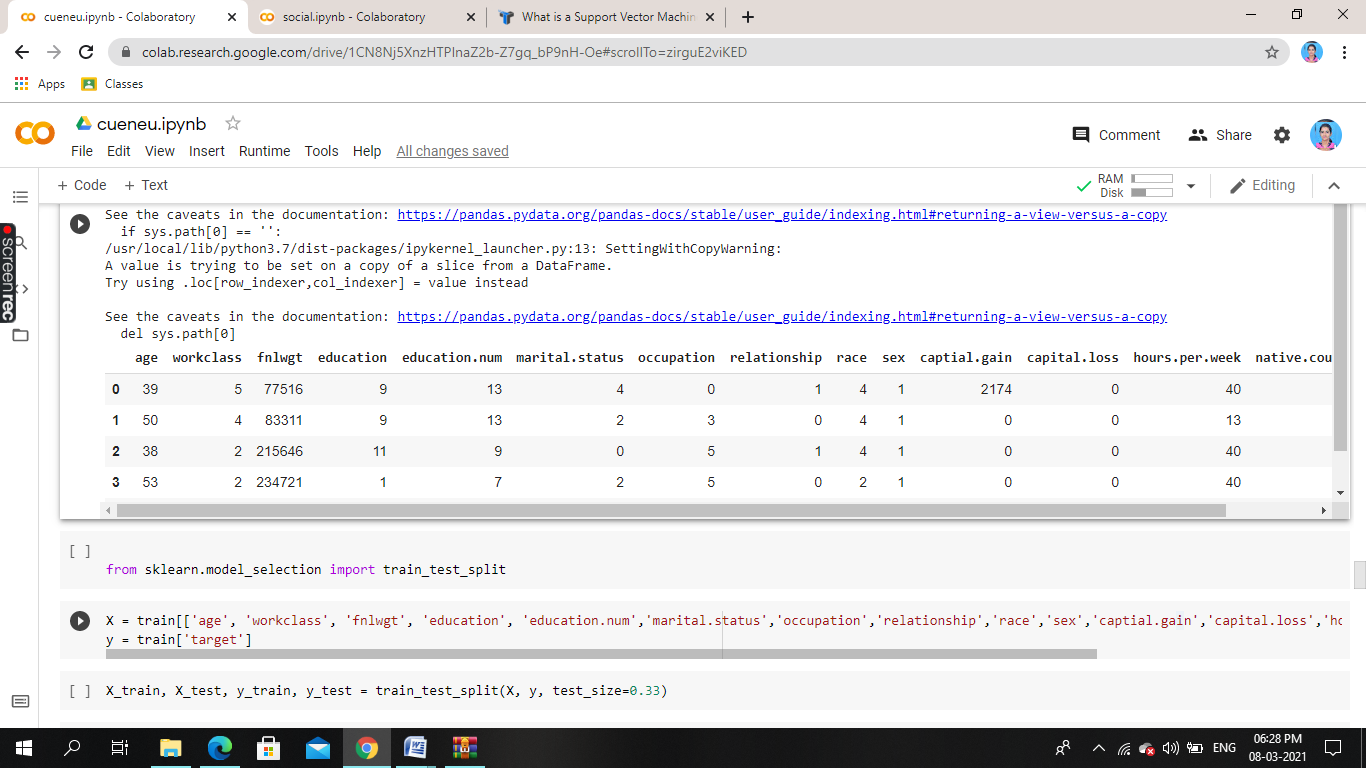
**INFO**

****

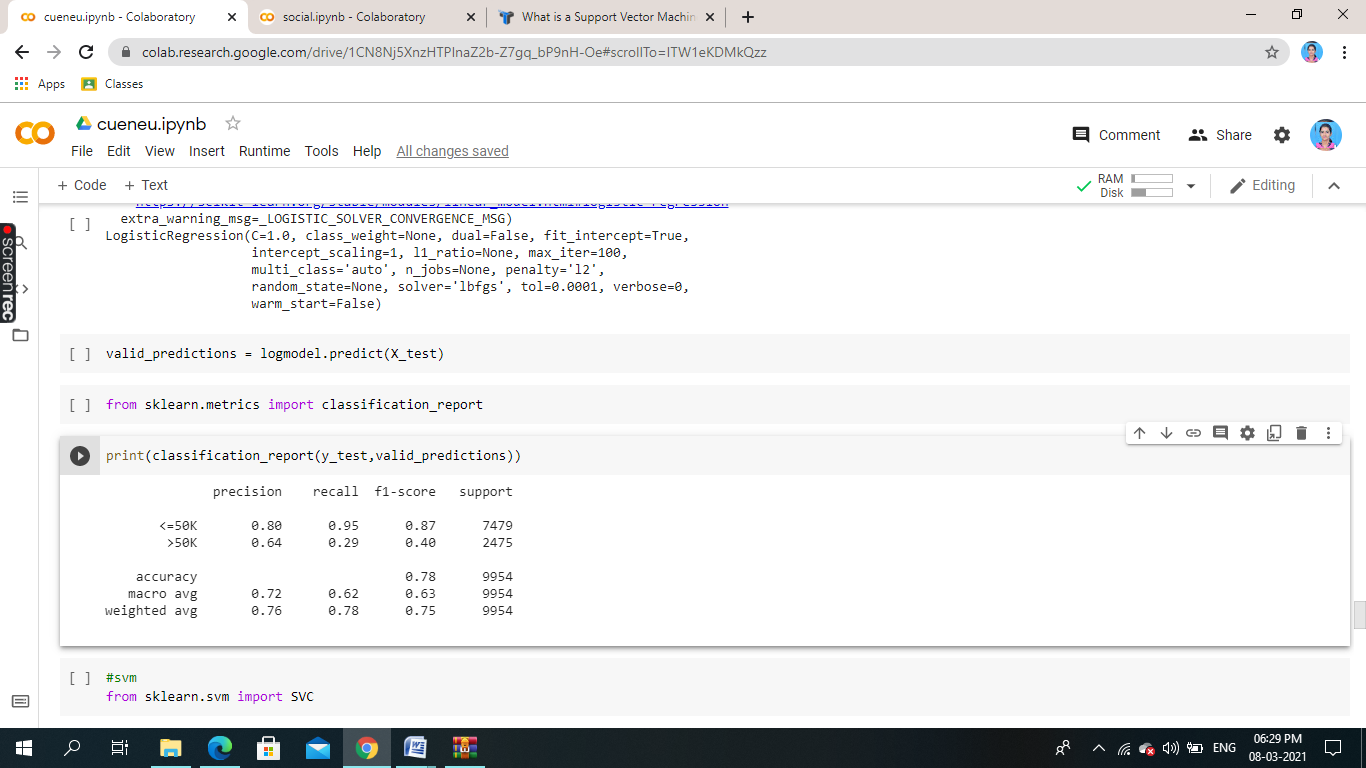
**DESCRIBE**

****

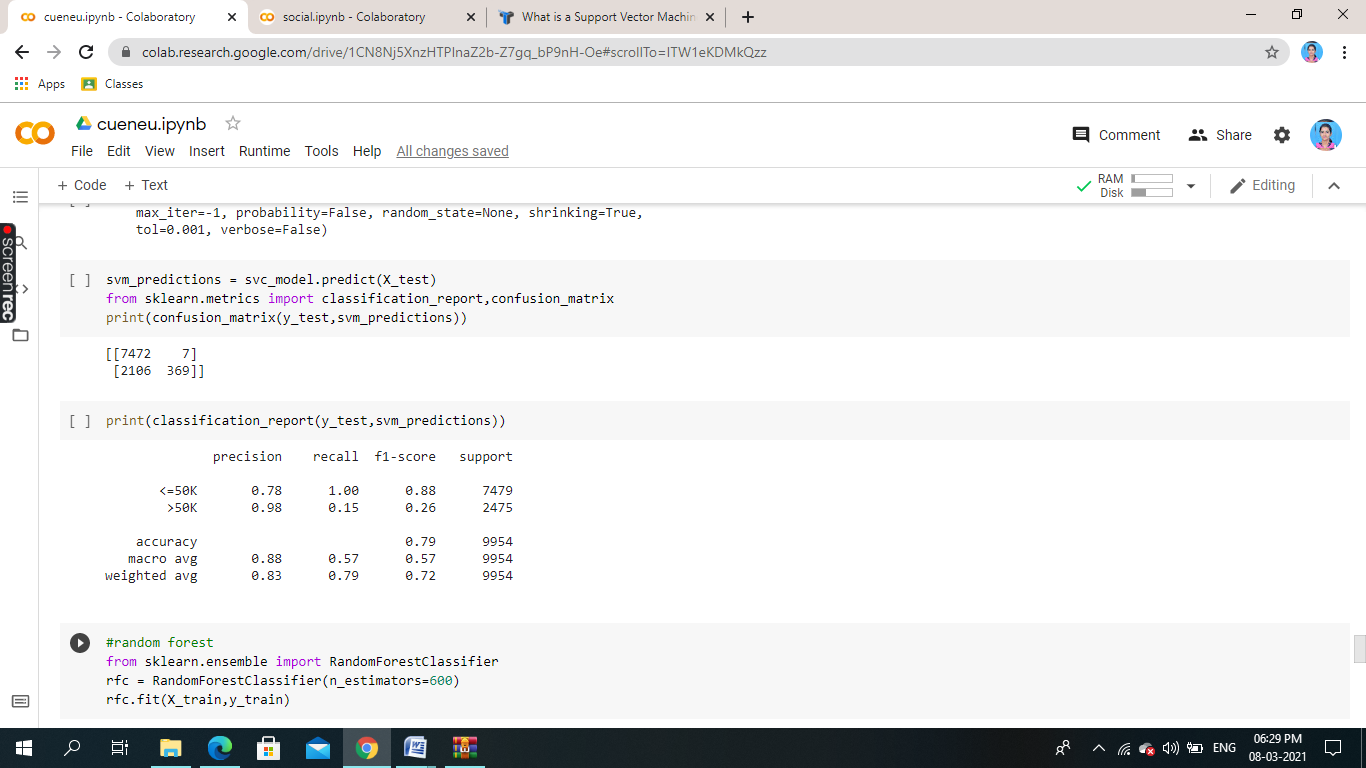
**ENCODED TEST VALUE**

****

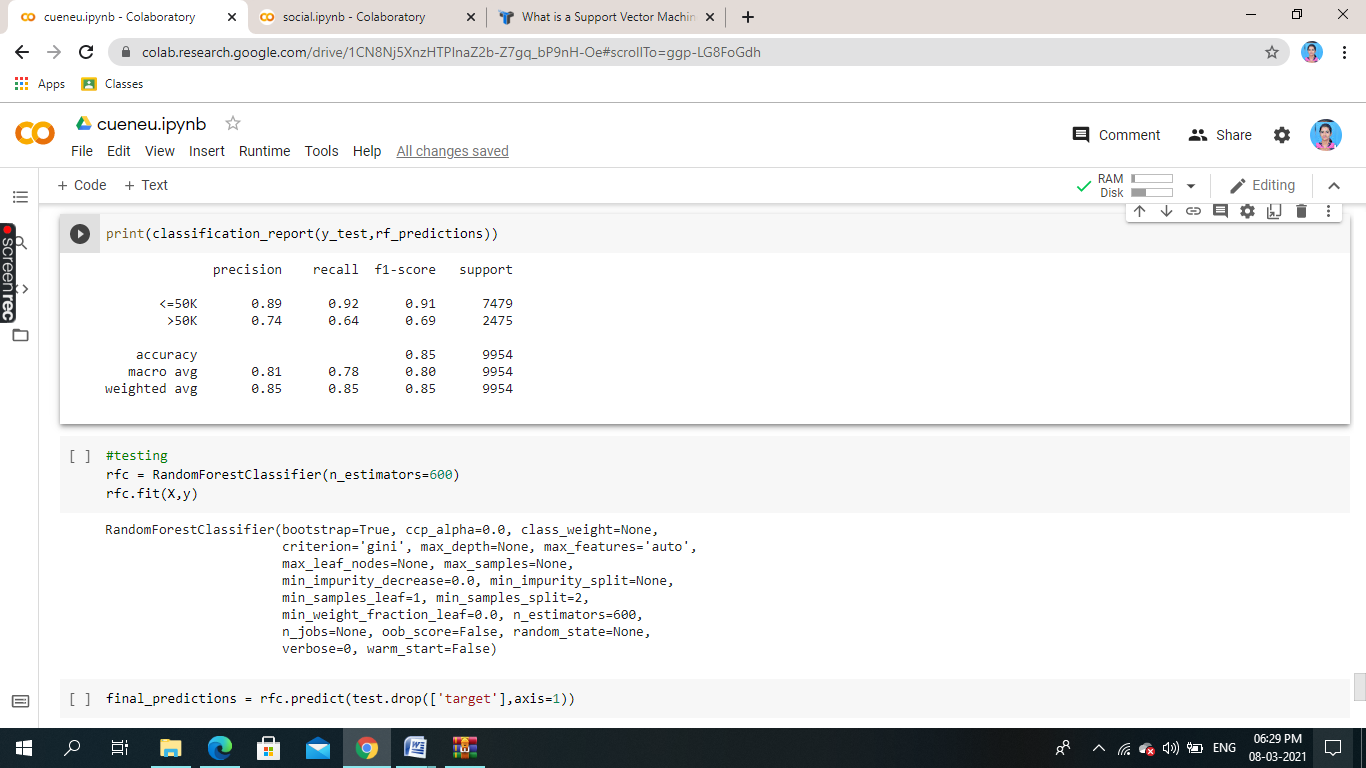
**LOGISTIC REGRESSION**

****

**SVM**

****

**RANDOM FOREST**

****

**CONCLUSION:**

Comparing the accuracy of the different models, Random forest is the best.

It gives accuracy of about 85% than compared to other to model.so random forest is better suited for this type of dataset.the salary of the person >50 or not are also shown by each type of model.