CURNEU MEDTECH INNOVATIONS PRIVATE LIMITED SD03Q08

-Kishore G (1832029)

Problem Statement:

To explore the dataset and build a suitable model for prediction whether the salary of the person is >50K or not and visualize the result and also provide with EDA techniques.

Abstract:

The main prospective of the problem is that to build a suitable machine learning model for the given salary dataset using Python. Logistic Regression is used here. For the given dataset, the missing data and noisy data are been compromised , analysed and finally output is obtained along with the Exploratory Data Analysis.

About the Dataset:

The dataset contains information on all the employees with their occupation, sex, race, work class etc. The last column of the dataset is describing whether or not the employee's salary is >50K or not. Both test and train datasets follows these conditions.

Test Dataset:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.	native.country	target
2	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
3	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spou	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
4	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
5	53	Private	234721	11th	7	Married-civ-spou	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
6	28	Private	338409	Bachelors	13	Married-civ-spou	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
7	37	Private	284582	Masters	14	Married-civ-spou	Exec-managerial	Wife	White	Female	0	0	40	United-States	<=50K
8	49	Private	160187	9th	5	Married-spouse-	Other-service	Not-in-family	Black	Female	0	0	16	Jamaica	<=50K
9	52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spou	Exec-managerial	Husband	White	Male	0	0	45	United-States	>50K
10	31	Private	45781	Masters	14	Never-married	Prof-specialty	Not-in-family	White	Female	14084	0	50	United-States	>50K
11	42	Private	159449	Bachelors	13	Married-civ-spou	Exec-managerial	Husband	White	Male	5178	0	40	United-States	>50K

Train Dataset:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week	native.country	target
2	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	United-States	<=50K.
3	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States	<=50K.
4	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States	>50K.
5	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	United-States	>50K.
6	18		103497	Some-college	10	Never-married		Own-child	White	Female	0	0	30	United-States	<=50K.
7	34	Private	198693	10th	6	Never-married	Other-service	Not-in-family	White	Male	0	0	30	United-States	<=50K.
8	29		227026	HS-grad	9	Never-married		Unmarried	Black	Male	0	0	40	United-States	<=50K.
9	63	Self-emp-not-inc	104626	Prof-school	15	Married-civ-spouse	Prof-specialty	Husband	White	Male	3103	0	32	United-States	>50K.
10	24	Private	369667	Some-college	10	Never-married	Other-service	Unmarried	White	Female	0	0	40	United-States	<=50K.
11	55	Private	104996	7th-8th	4	Married-civ-spouse	Craft-repair	Husband	White	Male	0	0	10	United-States	<=50K.

Logistic Regression:

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression).

Code:

```
# Import libraries

import numpy as np # linear algebra
import pandas as pd # data processing,

# Libraries for data visualization
import matplotlib.pyplot as pplt
import seaborn as sns
from pandas.plotting import scatter_matrix

# Import scikit_learn module for the algorithm/model: Linear Regression
from sklearn.linear_model import LogisticRegression
# Import scikit_learn module to split the dataset into train.test sub-datasets
from sklearn.model_selection import train_test_split
# Import scikit_learn module for k-fold cross validation
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
# import the metrics class
from sklearn import metrics
# import stats for accuracy
import statsmodels.api as sm

salary_dataset =
pd.read_csv(r'C:/Users/gkish/Downloads/SD03Q08/SD03Q08/SD03Q08/train.csv')
```

```
salary_dataset.info()


salary_dataset.rename(columns={'capital.gain': 'capital gain', 'capital.loss': 'capital loss',
'native.country': 'country','hours.per.week': 'hours per week','marital.status': 'marital'},
inplace=True)
salary_dataset.columns


salary_dataset.isin(['?']).sum(axis=0)


# code will replace the special character to nan and then drop the columns
salary_dataset['country'] = salary_dataset['country'].replace('?',np.nan)
salary_dataset['workclass'] =salary_dataset['workclass'].replace('?',np.nan)
salary_dataset['occupation'] =salary_dataset['occupation'].replace('?',np.nan)
#dropping the NaN rows now
salary_dataset.dropna(how='any',inplace=True)


#running a loop of value_counts of each column to find out unique values.
for c in salary_dataset.columns:
    print ("---- %s ---" % c)
    print (salary_dataset[c].value_counts())



#dropping based on uniquness of data from the dataset
salary_dataset.drop(['age', 'hours per week', 'fnlwgt', 'capital gain','capital loss', 'country'],
axis=1, inplace=True)
print(salary_dataset.head())



df = salary_dataset.replace({'sex':{' Male':0, ' Female':1},'target':{' <=50K':0, ' >50K':1}})
```

```
#race

df['race'] = df['race'].map({' Black': 0, ' Asian-Pac-Islander': 1, ' Other': 2, ' White': 3, ' Amer-Indian-Eskimo': 4}).astype(int)

#marital

df['marital'] = df['marital'].map({' Married-spouse-absent': 0, ' Widowed': 1, ' Married-civ-spouse': 2, ' Separated': 3, ' Divorced': 4,' Never-married': 5, ' Married-AF-spouse': 6}).astype(int)

#workclass

df['workclass'] = df['workclass'].map({' Self-emp-inc': 0, ' State-gov': 1,' Federal-gov': 2, ' Without-pay': 3, ' Local-gov': 4,' Private': 5, ' Self-emp-not-inc': 6}).astype(int)

#education

df['education'] = df['education'].map({' Some-college': 0, ' Preschool': 1, ' 5th-6th': 2, ' HS-grad': 3, ' Masters': 4, ' 12th': 5, ' 7th-8th': 6, ' Prof-school': 7,' 1st-4th': 8, ' Assoc-acdm': 9, ' Doctorate': 10, ' 11th': 11,' Bachelors': 12, ' 10th': 13,' Assoc-voc': 14,' 9th': 15}).astype(int)

#occupation

df['occupation'] = df['occupation'].map({ ' Farming-fishing': 1, ' Tech-support': 2, ' Adm-clerical': 3, ' Handlers-cleaners': 4,

 ' Prof-specialty': 5,' Machine-op-inspct': 6, ' Exec-managerial': 7,' Priv-house-serv': 8,' Craft-repair': 9,' Sales': 10, ' Transport-moving': 11, ' Armed-Forces': 12, ' Other-service': 13,' Protective-serv':14}).astype(int)

#relationship

df['relationship'] = df['relationship'].map({' Not-in-family': 0, ' Wife': 1, ' Other-relative': 2, ' Unmarried': 3,' Husband': 4,' Own-child': 5}).astype(int)

print(df.head())



#plotting a bar graph for Education against Income to see the co-relation between these columns

df.groupby('education').target.mean().plot(kind='bar')

df.groupby('occupation').target.mean().plot(kind='bar')

df.groupby('sex').target.mean().plot(kind='bar')

df.groupby('marital').target.mean().plot(kind='bar')

df.groupby('race').target.mean().plot(kind='bar')

df.groupby('workclass').target.mean().plot(kind='bar')
```

#Transform the data set into a data frame

#X axis = We concatenate the Relationship, Education,Race,Occupation columns concate using np.c_ provided by the numpy library

df_x = pd.DataFrame(np.c_[df['relationship'], df['education'], df['race'],df['occupation'],df['sex'],df['marital'],df['workclass']], columns = ['relationship','education','race','occupation','gender','marital','workclass'])

#Y axis = Our dependent variable or the income of adult i.e Income

df_y = pd.DataFrame(df.target)

#---

salary_dataset1 = pd.read_csv(r'C:/Users/gkish/Downloads/SD03Q08/SD03Q08/SD03Q08/test.csv')

salary_dataset1.info()

salary_dataset1.rename(columns={'capital.gain': 'capital gain', 'capital.loss': 'capital loss', 'native.country': 'country','hours.per.week': 'hours per week','marital.status': 'marital'}, inplace=True)

salary_dataset1.columns

salary_dataset1.isin(['?']).sum(axis=0)

code will replace the special character to nan and then drop the columns

salary_dataset1['country'] = salary_dataset1['country'].replace('?',np.nan)

salary_dataset1['workclass'] =salary_dataset1['workclass'].replace('?',np.nan)

salary_dataset1['occupation'] =salary_dataset1['occupation'].replace('?',np.nan)

#dropping the NaN rows now

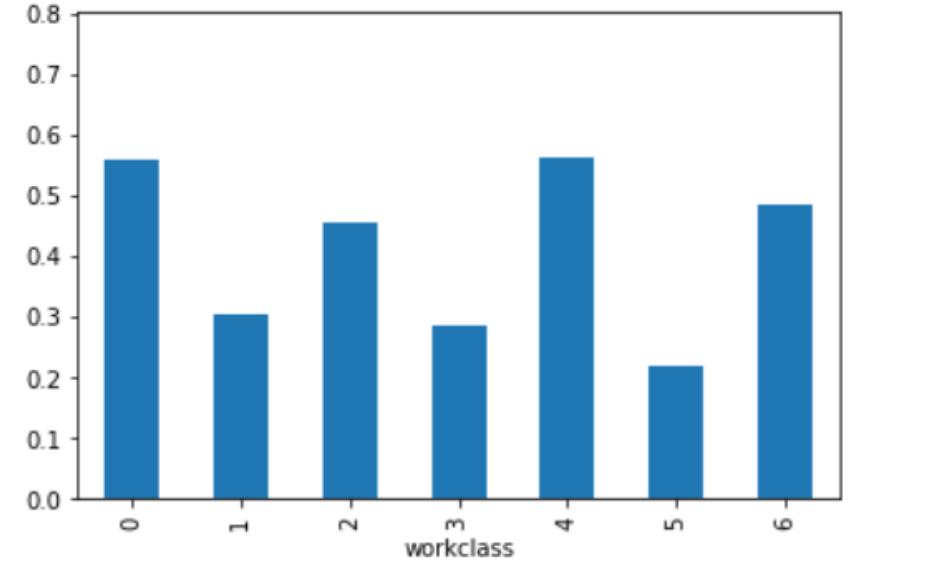
```
salary_dataset1.dropna(how='any',inplace=True)


#running a loop of value_counts of each column to find out unique values.
for c in salary_dataset1.columns:
    print ("---- %s ---" % c)
    print (salary_dataset1[c].value_counts())



#dropping based on uniquness of data from the dataset
salary_dataset1.drop(['age', 'hours per week', 'fnlwgt', 'capital gain','capital loss', 'country'], axis=1, inplace=True)
print(salary_dataset1.head())



df1 = salary_dataset1.replace({'sex':{' Male':0, ' Female':1},'target':{' <=50K.':0, ' >50K.':1}})



#race
df1['race'] = df1['race'].map({' Black': 0, ' Asian-Pac-Islander': 1, ' Other': 2, ' White': 3, ' Amer-Indian-Eskimo': 4}).astype(int)
#marital
df1['marital'] = df1['marital'].map({' Married-spouse-absent': 0, ' Widowed': 1, ' Married-civ-spouse': 2, ' Separated': 3, ' Divorced': 4,' Never-married': 5, ' Married-AF-spouse': 6}).astype(int)
#workclass
df1['workclass'] = df1['workclass'].map({' Self-emp-inc': 0, ' State-gov': 1,' Federal-gov': 2, ' Without-pay': 3, ' Local-gov': 4,' Private': 5, ' Self-emp-not-inc': 6}).astype(int)
#education
df1['education'] = df1['education'].map({' Some-college': 0, ' Preschool': 1, ' 5th-6th': 2, ' HS-grad': 3, ' Masters': 4, ' 12th': 5, ' 7th-8th': 6, ' Prof-school': 7,' 1st-4th': 8, ' Assoc-acdm': 9, ' Doctorate': 10, ' 11th': 11,' Bachelors': 12, ' 10th': 13,' Assoc-voc': 14,' 9th': 15}).astype(int)
#occupation
```

```
df1['occupation'] = df1['occupation'].map({ ' Farming-fishing': 1, ' Tech-support': 2, ' Adm-clerical': 3, ' Handlers-cleaners': 4,

 ' Prof-specialty': 5,' Machine-op-inspct': 6, ' Exec-managerial': 7,' Priv-house-serv': 8,' Craft-repair': 9,' Sales': 10, ' Transport-moving': 11, ' Armed-Forces': 12, ' Other-service': 13,' Protective-serv':14}).astype(int)

#relationship

df1['relationship'] = df1['relationship'].map({' Not-in-family': 0, ' Wife': 1, ' Other-relative': 2, ' Unmarried': 3,' Husband': 4,' Own-child': 5}).astype(int)

print(df1.head())



#plotting a bar graph for Education against Income to see the co-relation between these columns

df1.groupby('education').target.mean().plot(kind='bar')

df1.groupby('occupation').target.mean().plot(kind='bar')

df1.groupby('sex').target.mean().plot(kind='bar')

df1.groupby('marital').target.mean().plot(kind='bar')

df1.groupby('race').target.mean().plot(kind='bar')

df1.groupby('workclass').target.mean().plot(kind='bar')



#Transform the data set into a data frame

#X axis = We concatenate the Relationship, Education,Race,Occupation columns concate using np.c_ provided by the numpy library

df1_x = pd.DataFrame(np.c_[df1['relationship'], df1['education'], df1['race'],df1['occupation'],df1['sex'],df1['marital'],df1['workclass']], columns = ['relationship','education','race','occupation','gender','marital','workclass'])

#Y axis = Our dependent variable or the income of adult i.e Income

df1_y = pd.DataFrame(df1.target)


#-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

```
#Initialize the linear regression model
reg = LogisticRegression()



#Train our model with the training data
reg.fit(df_x, df_y)
#print our price predictions on our test data
y_pred = reg.predict(df1_x)



#feeding the predict function with our test values in the format
[['relationship','education','race','occupation','sex','marital','workclass']]
reg.predict([[1,7,3,7,0,2,0]])

#printing the accuracy values
print("Accuracy:",metrics.accuracy_score(df1_y, y_pred))
```

Output:

```
Accuracy: 0.7561752988047808
```



Conclusion:

Finally, the model has been obtained with an accuracy of 76% . But while there is always room for improvement, we can be satisfied with this model as our final product. Our accuracy is high, but not so high that we need to be suspicious of any over fitting. We can safely say that an increase in Estimated Salary will lead to a higher probability.