Income Classification Problem Solution

1. Importing necessary modules to work on this problem

```
import pandas as pd                                          # To work with dataframes
import numpy as np                                           # To Perform numerical operations
import seaborn as sns                                        #To visualize data
from sklearn.model_selection import train_test_split         #To Partition the data
from sklearn.linear_model import LogisticRegression          #Importing Library for Logistic
                                                             regression
from sklearn.metrics import accuracy_score,confusion_matrix  #Importing Performance matrix
```

2. To read the csv file

   data_income = pd.read_csv("income.csv")

3. Create a Copy to protect original data

   data = data_income.copy()

   here we will work on the same dataset which is clone of original dataset.

4. Data Pre-processing - Missing values

   data.info() *#To get the data types*

   Gives Column;Non-Null Count; Data type (int, string, object etc)

5. *data.isnull()* #Check for missing values

   Also *data.isnull().sum()* gives sum of columns that are having null values.

6. *summary_num = data.describe()* #gives the mean, std, and min, max values

7. *summary_cate = data.describe(include="O")* #includes objects
   summary_cate #gives unique, most frequently occuring category etc

8. Get all the Unique categories using *value_counts*

   data['JobType'].value_counts()

9. *np.unique(data['JobType'])* # np.unique gives unique categories

   This is to remove categories like ' ?' or ' ??' etc which are irrevalent.

10. Check if in a particular row either one of the column value is missing or both are missing under Job type and Occupation. lets subset the row with atleast one column missing in a row. axis=1 gives atleast one missing column.

    missing = data[data.isnull().any(axis=1)]

11. We see that it has converted all " ?" values to nan. But we found some datasets(7 Nos) wehere the Jobtype is Never Worked and therefore occupation is given as nan

    Drop rows with missing values
    data2 = data.dropna(axis=0)

12. *correlation = data2.corr()* #Find out correlation between variables.
    correlation #only showing correlation betwen numerical variables

13. Now we will consider categorical variables
    The pandas crosstab function builds a cross-tabulation table that can show the frequency with which certain groups of data appear
    #gender proportion table
    gender = pd.crosstab(index=data2['gender'],columns='count',normalize=True)
    gender
    ex:

col_0	count
gender	
Female	0.324315
Male	0.675685

14. Above if we set normalize='index' we get row proportion=1
    Salstat = sns.countplot(x=data2['gender'])

15. Histogram of Age
    sns.displot(x=data2['age'],bins=5,kde=False)

    bins = No of bars or subdivisions
    kde = False, True means we need distribution curve or not.

16. Boxplot to see the outliers

    sns.boxplot(x='SalStat',y='age',data=data2)

17. Group by is used to group categories

    ata2.groupby('SalStat')['age'].median()

    This command groups by age the salary status.

    Example:

    SalStat
    greater than 50,000 43
    less than or equal to 50,000 34

    which means
    People with 35 to 50 Age group are more likely to earn > 50000
    People with 25 to 34 Age group are more likely to earn < 50000

    Logisitc regression

18. Reindexing the salary status names to 0,1 Because machine Learning data cannot work with categorical data directly. So categorical data must be converted to numbers.

    data2['SalStat'] = data2['SalStat'].map({' less than or equal to 50,000':0,' greater than 50,000':1})

    Less than equal to 50000 becomes zero and more than equals to becomes one.

19. Replace nan values with zero
    data2['SalStat'] = data2['SalStat'].replace(np.nan, 0)

20. Convert categorical variable into dummy or indicator variables

    new_data = pd.get_dummies(data2,drop_first=True)

21. Storing only the Column names

    column_list = list(new_data.columns)

22. Separating the input names from data

    features = list(set(column_list) - set(['SalStat']))

23. storing the output values in y

    y=new_data['SalStat'].values

    Storing the values from input features. For features dont use single quotes new_data['features'] like above else you will get error KeyError: For 'features'. just use new_data[features].values

    x= new_data[features].values

24. *x.ndim* will give number of dimensions of x.

25. Splitting the data into train and test

    train_x, test_x, train_y, test_y = train_test_split(x,y,test_size=0.3,random_state=0)

    test_size decides what proportion will be test data and *random_state=0* means same set of samples will be chosen, if not given then different set of samples will be chosen.

26. Make an instance of the Model

    logistic = LogisticRegression()

    If it do not converge and you get an error

    ConvergenceWarning: lbfgs failed to converge (status=1):STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

    Then use iter=10000 as a parameter that says how many number of iterations we need.

    logistic = LogisticRegression(max_iter=10000)

27. Fitting the values for x and y

    logistic.fit(train_x,train_y)

28. Getting Coefficient and Intercept of the result

    *logistic.coef_*

*logistic.intercept_*

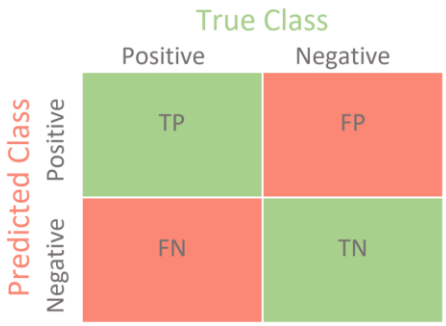
29. Predicting the test values

    prediction1 = logistic.predict(test_x)
    print(prediction1)

30. Confusion matrix to get the True and False positive

    confusion_matrix1 = confusion_matrix(test_y, prediction1)
    print(confusion_matrix1)

    We get a result like this



[[6317 506]
[927 1299]]

True Positive – 6317
False Positive -506
False Negative – 927
True Negative - 1299

31. Calculating the Accuracy

    accuracy_score1 = accuracy_score(test_y,prediction1)
    print(accuracy_score1)

32. Printing the misclassified value from Prediction

    print('Misclassified samples: %d' % (test_y != prediction).sum())

33. Logistic Regression - Removing Insignificant variables

    cols = ['gender','nativecountry','race','JobType']
    new_data = data2.drop(cols,axis = 1)
    new_data = pd.get_dummies(new_data,drop_first=True)

KNN Classifier

34. Importing the module

    from sklearn.neighbors import KNeighborsClassifier

35. import library for plotting

    import matplotlib.pyplot as plt

36. Storing the K nearest neighbors classifier

    KNN_Classifier = KNeighborsClassifier(n_neighbors=5)

37. fitting the values for x and y

    KNN_Classifier.fit(train_x, train_y)

38. predicting the text values with model

    prediction = KNN_Classifier.predict(test_x)

39. Performance metric check

    confusion_matrix = confusion_matrix(test_y,prediction)
    print("\t","Predicted Values")
    print("Original values","\n",confusion_matrix)

40. calculating the accuracy

    accuracy_score3 = accuracy_score(test_y, prediction)
    print(accuracy_score3)

41. *print('Misclassified samples: %d' %(test_y != prediction).sum())*

42. Another way to check misclassified samples is

    Misclassified_sample = []
    #Claculating error for k values between 1 ans 20
    for i in range(1,20):

```
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(train_x, train_y)
    pred_i = knn.predict(test_x)
    Misclassified_sample.append((test_y != pred_i).sum())

print(Misclassified_sample)
```

n_neighbors=10 it is lowest