Project Development

Delivery Of Sprint-3

Date	03 October 2022
Team ID	PNT2022TMID37447
Project Name	Project - Corporate Employee Attrition Analytics

DATA UNDERSTANDING, DATA PREPARATION & TESTING

CATEGORICAL VARIABLES CORRELATION

HYPOTHESIS TESTING CONDITIONS

Our hypotheses will be:

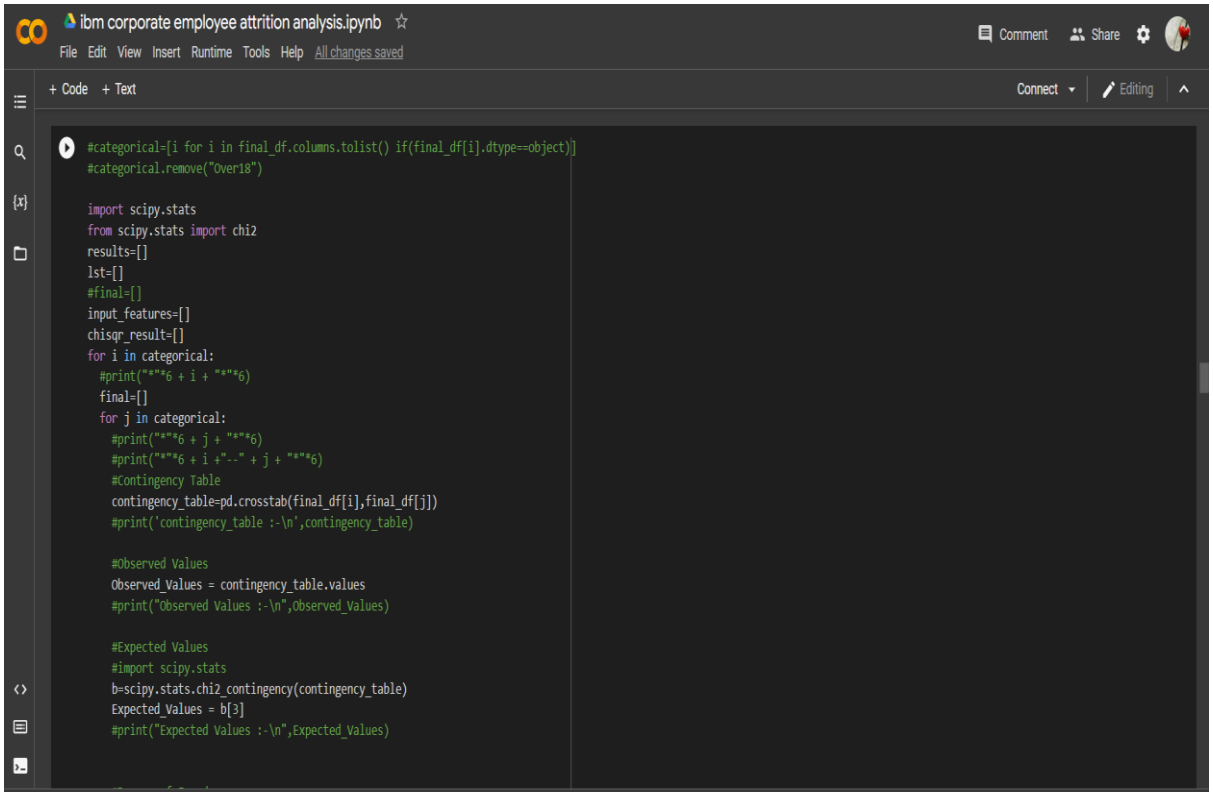
- Null Hypothesis (H0)

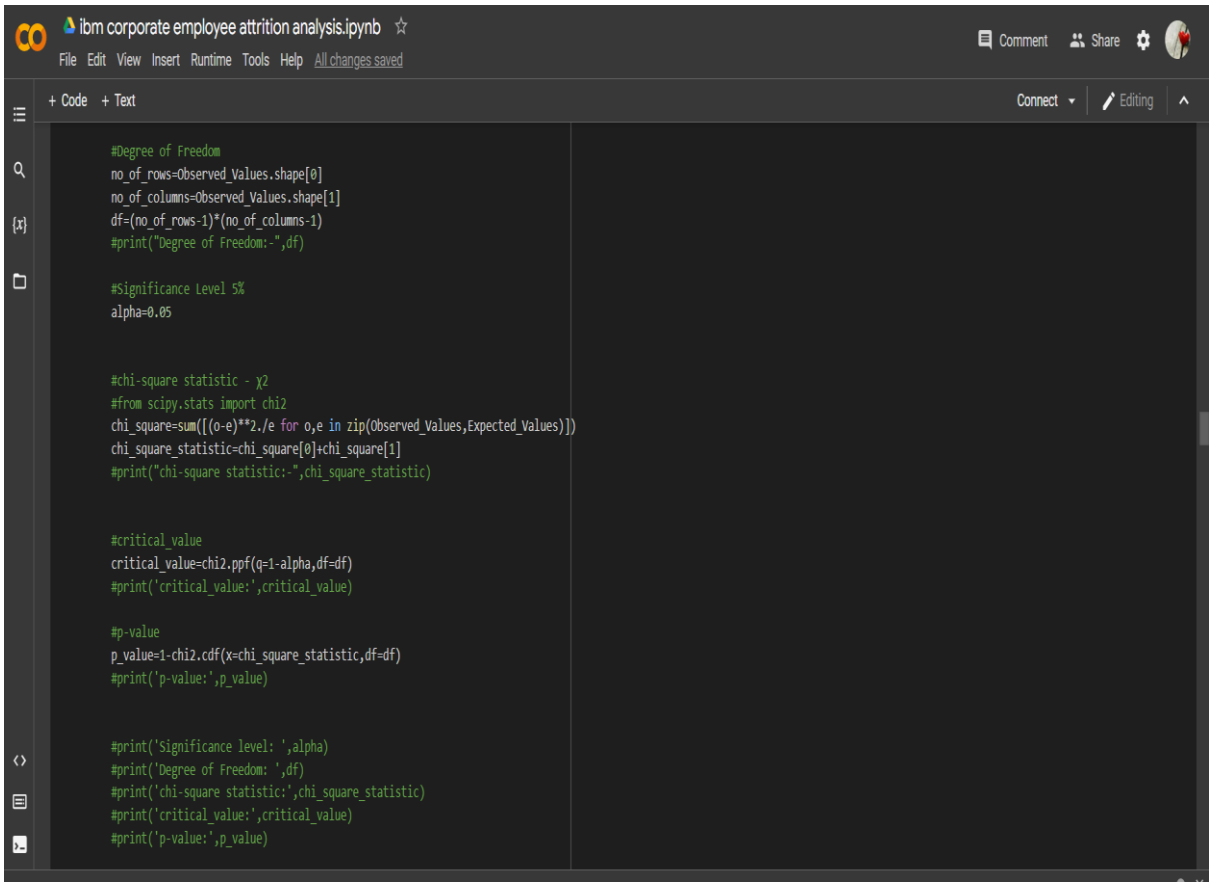
H0: There is no relationship between 2 categorial variables ie Both features or variables are independent of each other

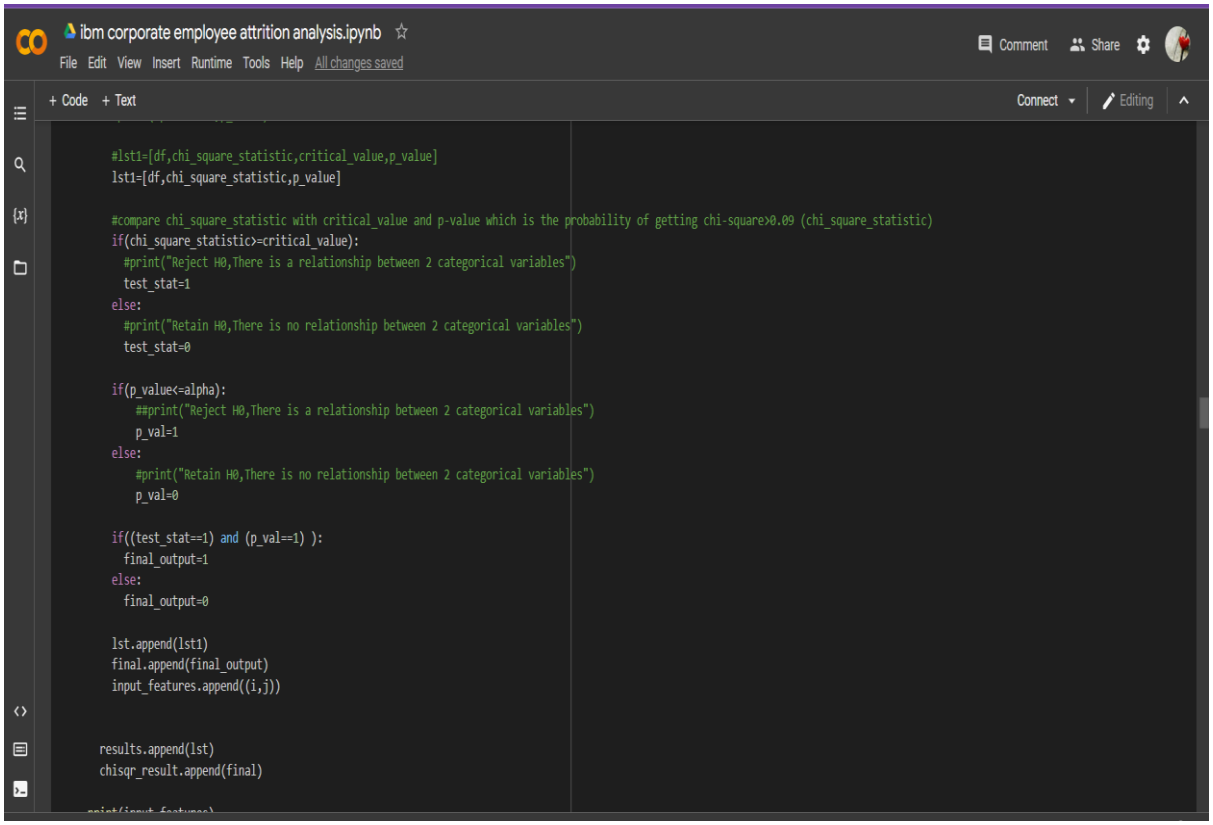
- Alternate Hypothesis (H1)

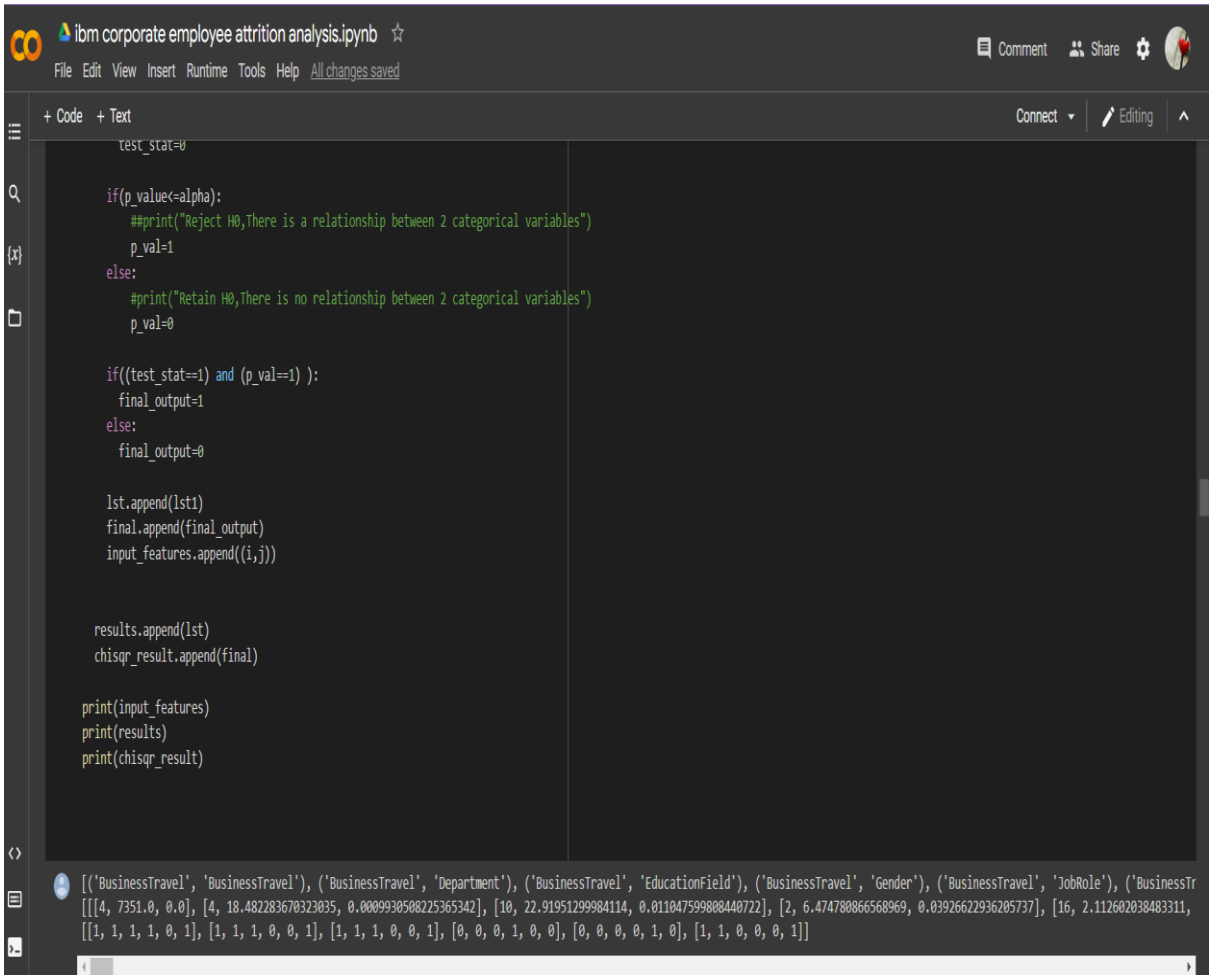
H1: There is Relationship between 2 categorical variables .ie Both features or variables are independent of each other

CODING:


ibm corporate employee attrition analysis.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Connect Editing
#categorical=[i for i in final_df.columns.tolist() if(final_df[i].dtype==object)]
#categorical.remove("Over18")
import scipy.stats
from scipy.stats import chi2
results=[]
lst=[]
#final=[]
input_features=[]
chisqr_result=[]
for i in categorical:
#print("*"*6 + i + "*"*6)
final=[]
for j in categorical:
#print("*"*6 + j + "*"*6)
#print("*"*6 + i +"--" + j + "*"*6)
#Contingency Table
contingency_table=pd.crosstab(final_df[i],final_df[j])
#print('contingency_table :-\n',contingency_table)
#Observed Values
Observed_Values = contingency_table.values
#print("Observed Values :-\n",Observed_Values)
#Expected Values
#import scipy.stats
b=scipy.stats.chi2_contingency(contingency_table)
Expected_Values = b[3]
#print("Expected Values :-\n",Expected_Values)


ibm corporate employee attrition analysis.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Connect Editing
#Degree of Freedom
no_of_rows=Observed_Values.shape[0]
no_of_columns=Observed_Values.shape[1]
df=(no_of_rows-1)*(no_of_columns-1)
#print("Degree of Freedom:-",df)
#Significance Level 5%
alpha=0.05
#chi-square statistic - χ2
#from scipy.stats import chi2
chi_square=sum([(o-e)**2./e for o,e in zip(Observed_Values,Expected_Values)])
chi_square_statistic=chi_square[0]+chi_square[1]
#print("chi-square statistic:-",chi_square_statistic)
#critical_value
critical_value=chi2.ppf(q=1-alpha,df=df)
#print('critical_value:',critical_value)
#p-value
p_value=1-chi2.cdf(x=chi_square_statistic,df=df)
#print('p-value:',p_value)
#print('Significance level: ',alpha)
#print('Degree of Freedom: ',df)
#print('chi-square statistic:',chi_square_statistic)
#print('critical_value:',critical_value)
#print('p-value:',p_value)


ibm corporate employee attrition analysis.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Connect
Editing
#lst1=[df,chi_square_statistic,critical_value,p_value]
lst1=[df,chi_square_statistic,p_value]
#compare chi_square_statistic with critical_value and p-value which is the probability of getting chi-square>0.09 (chi_square_statistic)
if(chi_square_statistic>=critical_value):
#print("Reject H0,There is a relationship between 2 categorical variables")
test_stat=1
else:
#print("Retain H0,There is no relationship between 2 categorical variables")
test_stat=0
if(p_value<=alpha):
##print("Reject H0,There is a relationship between 2 categorical variables")
p_val=1
else:
#print("Retain H0,There is no relationship between 2 categorical variables")
p_val=0
if((test_stat==1) and (p_val==1)):
final_output=1
else:
final_output=0
lst.append(lst1)
final.append(final_output)
input_features.append((i,j))
results.append(lst)
chisqr_result.append(final)


ibm corporate employee attrition analysis.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Connect
Editing
test_stat=0
if(p_value<=alpha):
##print("Reject H0,There is a relationship between 2 categorical variables")
p_val=1
else:
#print("Retain H0,There is no relationship between 2 categorical variables")
p_val=0
if((test_stat==1) and (p_val==1)):
final_output=1
else:
final_output=0
lst.append(lst1)
final.append(final_output)
input_features.append((i,j))
results.append(lst)
chisqr_result.append(final)
print(input_features)
print(results)
print(chisqr_result)
[('BusinessTravel', 'BusinessTravel'), ('BusinessTravel', 'Department'), ('BusinessTravel', 'EducationField'), ('BusinessTravel', 'Gender'), ('BusinessTravel', 'JobRole'), ('BusinessTr
[[[4, 7351.0, 0.0], [4, 18.482283670323035, 0.000993050822536534 2], [10, 22.91951299984114, 0.011047599808440722], [2, 6.4747808665688969, 0.03926622936205737], [16, 2.112602038483311,
[[1, 1, 1, 1, 0, 1], [1, 1, 1, 0, 0, 1], [1, 1, 1, 0, 0, 1], [0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 1, 0], [1, 1, 0, 0, 0, 1]]

```
#categorical=[i for i in final_df.columns.tolist() if(final_df[i].dtype==object)]
#categorical.remove("Over18")

import scipy.stats
from scipy.stats import chi2
results=[]
lst=[]
#final=[]
input_features=[]
chisqr_result=[]
for i in categorical:
  #print("*"*6 + i + "*"*6)
  final=[]
  for j in categorical:
    #print("*"*6 + j + "*"*6)
    #print("*"*6 + i +"--" + j + "*"*6)
    #Contingency Table
    contingency_table=pd.crosstab(final_df[i],final_df[j])
    #print('contingency_table :-\n',contingency_table)

    #Observed Values
    Observed_Values = contingency_table.values
    #print("Observed Values :-\n",Observed_Values)

    #Expected Values
    #import scipy.stats
```

```
b=scipy.stats.chi2_contingency(contingency_table)
Expected_Values = b[3]
#print("Expected Values :-\n",Expected_Values)

#Degree of Freedom
no_of_rows=Observed_Values.shape[0]
no_of_columns=Observed_Values.shape[1]
df=(no_of_rows-1)*(no_of_columns-1)
#print("Degree of Freedom:-",df)

#Significance Level 5%
alpha=0.05

#chi-square statistic - χ2
#from scipy.stats import chi2
chi_square=sum([(o-
e)**2./e for o,e in zip(Observed_Values,Expected_Values)])
chi_square_statistic=chi_square[0]+chi_square[1]
#print("chi-square statistic:-",chi_square_statistic)

#critical_value
critical_value=chi2.ppf(q=1-alpha,df=df)
#print('critical_value:',critical_value)

#p-value
p_value=1-chi2.cdf(x=chi_square_statistic,df=df)
#print('p-value:',p_value)
```

```
    #print('Significance level: ',alpha)
    #print('Degree of Freedom: ',df)
    #print('chi-square statistic:',chi_square_statistic)
    #print('critical_value:',critical_value)
    #print('p-value:',p_value)

    #lst1=[df,chi_square_statistic,critical_value,p_value]
    lst1=[df,chi_square_statistic,p_value]

    #compare chi_square_statistic with critical_value and p-
value which is the probability of getting chi-square>0.09 (chi_square_statistic)
    if(chi_square_statistic>=critical_value):
      #print("Reject H0,There is a relationship between 2 categorical variables")
      test_stat=1
    else:
      #print("Retain H0,There is no relationship between 2 categorical variables")
      test_stat=0

    if(p_value<=alpha):
       ##print("Reject H0,There is a relationship between 2 categorical variables")
       p_val=1
    else:
       #print("Retain H0,There is no relationship between 2 categorical variables")
       p_val=0

    if((test_stat==1) and (p_val==1) ):
```

```
        final_output=1
    else:
        final_output=0

    lst.append(lst1)
    final.append(final_output)
    input_features.append((i,j))


  results.append(lst)
  chisqr_result.append(final)

print(input_features)
print(results)
print(chisqr_result)
```

OUTPUT:

[('BusinessTravel', 'BusinessTravel'), ('BusinessTravel', 'Department'), ('BusinessTravel', 'EducationField'), ('BusinessTravel', 'Gender'), ('BusinessTravel', 'JobRole'), ('BusinessTravel', 'MaritalStatus'), ('Department', 'BusinessTravel'), ('Department', 'Department'), ('Department', 'EducationField'), ('Department', 'Gender'), ('Department', 'JobRole'), ('Department', 'MaritalStatus'), ('EducationField', 'BusinessTravel'), ('EducationField', 'Department'), ('EducationField', 'EducationField'), ('EducationField', 'Gender'), ('EducationField', 'JobRole'), ('EducationField', 'MaritalStatus'), ('Gender', 'BusinessTravel'), ('Gender', 'Department'), ('Gender', 'EducationField'), ('Gender', 'Gender'), ('Gender', 'JobRole'), ('Gender', 'MaritalStatus'), ('JobRole', 'BusinessTravel'), ('JobRole', 'Department'), ('JobRole', 'EducationField'), ('JobRole', 'Gender'), ('JobRole', 'JobRole'), ('JobRole', 'MaritalStatus'), ('MaritalStatus', 'BusinessTravel'), ('MaritalStatus', 'Department'), ('MaritalStatus', 'EducationField'), ('MaritalStatus', 'Gender'), ('MaritalStatus', 'JobRole'), ('MaritalStatus', 'MaritalStatus')]
[[[4, 7351.0, 0.0], [4, 18.482283670323035, 0.0009930508225365342], [10, 22.91951299984114, 0.011047599808440722], [2, 6.474780866568969, 0.03926622936205737], [16, 2.112602038483311, 0.9999848809548781], [4, 19.96730541356567, 0.0005068757482377118], [4, 24.107877157800974, 7.599456876472566e-05], [4, 5606.999999999999, 0.0], [10, 1814.9940082404353, 0.0], [2, 0.6554237083518072, 0.7205706192092916], [16, 7.0608955020156134, 0.9720693962380044], [4, 21.01893128218694, 0.0003139440272663663], [10, 43.21521906368869, 4.55116138065037e-06], [10, 2149.140329473462, 0.0], [25, 6754.0, 0.0], [5, 8.173363631380072, 0.1469364827306855], [40, 16.27224873151621, 0.9996876232608488], [10, 21.99437742174664, 0.015133269763218382], [2, 5.46509022138302, 0.06505351040946972], [2, 0.6156417705991506, 0.7350469639355997], [5,

3.586949392393711, 0.6102738423265935], [1, 4300.0, 0.0], [8, 0.10167285358473775, 0.9999997327969251], [2, 3.979090758630589, 0.13675758414167993], [16, 21.346364236257617, 0.16557767933875145], [16, 21.87562551229072, 0.1472530084755297], [40, 21.445350078938674, 0.9928153627256894], [8, 12.642109094561857, 0.12477119469827436], [64, 8069.0, 0.0], [16, 23.660196523911065, 0.09719334311698913], [4, 16.929818667528195, 0.001994587277967308], [4, 15.257701399909378, 0.004195514172212866], [10, 12.780313848683722, 0.23621543026603953], [2, 4.206509895247947, 0.12205848698313337], [16, 11.408404447909351, 0.7836029365000337], [4, 5682.0, 0.0]], [[4, 7351.0, 0.0], [4, 18.482283670323035, 0.0009930508225365342], [10, 22.91951299984114, 0.011047599808440722], [2, 6.474780866568969, 0.03926622936205737], [16, 2.112602038483311, 0.9999848809548781], [4, 19.96730541356567, 0.0005068757482377118], [4, 24.107877157800974, 7.599456876472566e-05], [4, 5606.999999999999, 0.0], [10, 1814.9940082404353, 0.0], [2, 0.6554237083518072, 0.7205706192092916], [16, 7.0608955020156134, 0.9720693962380044], [4, 21.01893128218694, 0.0003139440272663663], [10, 43.21521906368869, 4.55116138065037e-06], [10, 2149.140329473462, 0.0], [25, 6754.0, 0.0], [5, 8.173363631380072, 0.1469364827306855], [40, 16.27224873151621, 0.9996876232608488], [10, 21.99437742174664, 0.015133269763218382], [2, 5.46509022138302, 0.06505351040946972], [2, 0.6156417705991506, 0.7350469639355997], [5, 3.586949392393711, 0.6102738423265935], [1, 4300.0, 0.0], [8, 0.10167285358473775, 0.9999997327969251], [2, 3.979090758630589, 0.13675758414167993], [16, 21.346364236257617, 0.16557767933875145], [16, 21.87562551229072, 0.1472530084755297], [40, 21.445350078938674, 0.9928153627256894], [8, 12.642109094561857, 0.12477119469827436], [64, 8069.0, 0.0], [16, 23.660196523911065, 0.09719334311698913], [4, 16.929818667528195, 0.001994587277967308], [4, 15.257701399909378, 0.004195514172212866], [10, 12.780313848683722, 0.23621543026603953], [2, 4.206509895247947, 0.12205848698313337], [16, 11.408404447909351, 0.7836029365000337], [4, 5682.0, 0.0]], [[4, 7351.0, 0.0], [4, 18.482283670323035, 0.0009930508225365342], [10, 22.91951299984114, 0.011047599808440722], [2, 6.474780866568969, 0.03926622936205737], [16, 2.112602038483311, 0.9999848809548781], [4, 19.96730541356567, 0.0005068757482377118], [4, 24.107877157800974, 7.599456876472566e-05], [4, 5606.999999999999, 0.0], [10, 1814.9940082404353, 0.0], [2, 0.6554237083518072, 0.7205706192092916], [16, 7.0608955020156134, 0.9720693962380044], [4, 21.01893128218694, 0.0003139440272663663], [10, 43.21521906368869, 4.55116138065037e-06], [10, 2149.140329473462, 0.0], [25, 6754.0, 0.0], [5, 8.173363631380072, 0.1469364827306855], [40, 16.27224873151621, 0.9996876232608488], [10, 21.99437742174664, 0.015133269763218382], [2, 5.46509022138302, 0.06505351040946972], [2, 0.6156417705991506, 0.7350469639355997], [5, 3.586949392393711, 0.6102738423265935], [1, 4300.0, 0.0], [8, 0.10167285358473775, 0.9999997327969251], [2, 3.979090758630589, 0.13675758414167993], [16, 21.346364236257617, 0.16557767933875145], [16, 21.87562551229072, 0.1472530084755297], [40, 21.445350078938674, 0.9928153627256894], [8, 12.642109094561857, 0.12477119469827436], [64, 8069.0, 0.0], [16, 23.660196523911065, 0.09719334311698913], [4, 16.929818667528195, 0.001994587277967308], [4, 15.257701399909378, 0.004195514172212866], [10, 12.780313848683722, 0.23621543026603953], [2, 4.206509895247947, 0.12205848698313337], [16, 11.408404447909351, 0.7836029365000337], [4, 5682.0, 0.0]], [[4, 7351.0, 0.0], [4, 18.482283670323035, 0.0009930508225365342], [10, 22.91951299984114, 0.011047599808440722], [2, 6.474780866568969, 0.03926622936205737], [16, 2.112602038483311, 0.9999848809548781], [4,

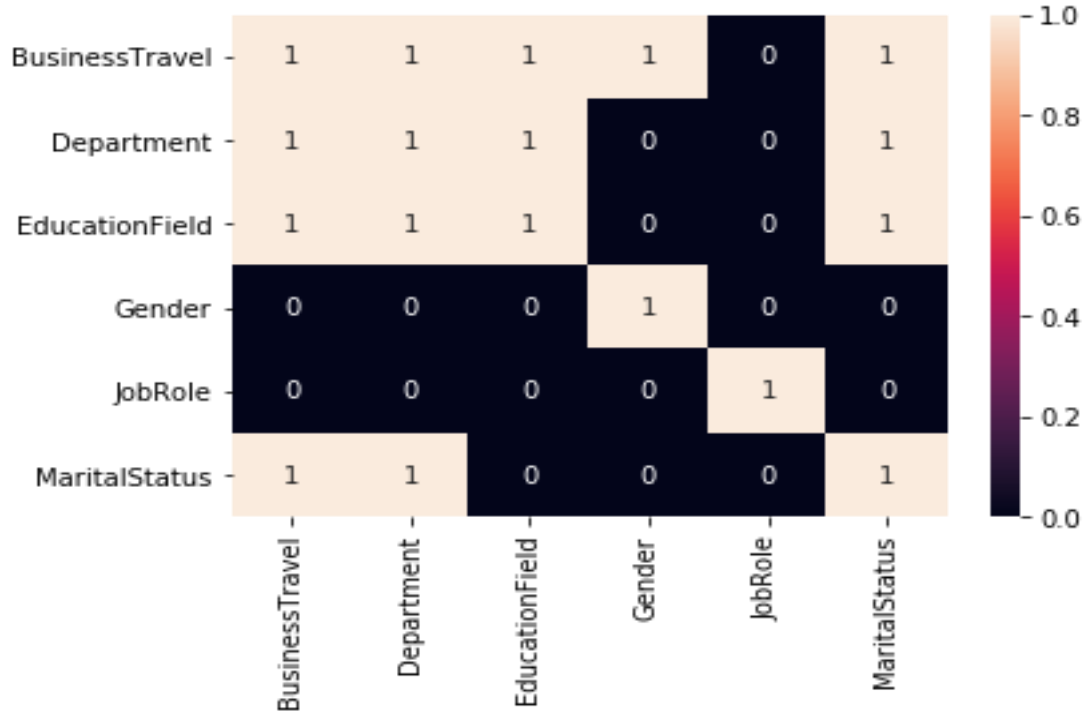
19.96730541356567, 0.0005068757482377118], [4, 24.107877157800974, 7.599456876472566e-05], [4, 5606.999999999999, 0.0], [10, 1814.9940082404353, 0.0], [2, 0.6554237083518072, 0.7205706192092916], [16, 7.0608955020156134, 0.9720693962380044], [4, 21.01893128218694, 0.0003139440272663663], [10, 43.21521906368869, 4.55116138065037e-06], [10, 2149.140329473462, 0.0], [25, 6754.0, 0.0], [5, 8.173363631380072, 0.1469364827306855], [40, 16.27224873151621, 0.9996876232608488], [10, 21.99437742174664, 0.015133269763218382], [2, 5.46509022138302, 0.06505351040946972], [2, 0.6156417705991506, 0.7350469639355997], [5, 3.586949392393711, 0.6102738423265935], [1, 4300.0, 0.0], [8, 0.10167285358473775, 0.9999997327969251], [2, 3.979090758630589, 0.13675758414167993], [16, 21.346364236257617, 0.16557767933875145], [16, 21.87562551229072, 0.1472530084755297], [40, 21.445350078938674, 0.9928153627256894], [8, 12.642109094561857, 0.12477119469827436], [64, 8069.0, 0.0], [16, 23.660196523911065, 0.09719334311698913], [4, 16.929818667528195, 0.001994587277967308], [4, 15.257701399909378, 0.004195514172212866], [10, 12.780313848683722, 0.23621543026603953], [2, 4.206509895247947, 0.12205848698313337], [16, 11.408404447909351, 0.7836029365000337], [4, 5682.0, 0.0]], [[4, 7351.0, 0.0], [4, 18.482283670323035, 0.0009930508225365342], [10, 22.91951299984114, 0.011047599808440722], [2, 6.474780866568969, 0.03926622936205737], [16, 2.112602038483311, 0.9999848809548781], [4, 19.96730541356567, 0.0005068757482377118], [4, 24.107877157800974, 7.599456876472566e-05], [4, 5606.999999999999, 0.0], [10, 1814.9940082404353, 0.0], [2, 0.6554237083518072, 0.7205706192092916], [16, 7.0608955020156134, 0.9720693962380044], [4, 21.01893128218694, 0.0003139440272663663], [10, 43.21521906368869, 4.55116138065037e-06], [10, 2149.140329473462, 0.0], [25, 6754.0, 0.0], [5, 8.173363631380072, 0.1469364827306855], [40, 16.27224873151621, 0.9996876232608488], [10, 21.99437742174664, 0.015133269763218382], [2, 5.46509022138302, 0.06505351040946972], [2, 0.6156417705991506, 0.7350469639355997], [5, 3.586949392393711, 0.6102738423265935], [1, 4300.0, 0.0], [8, 0.10167285358473775, 0.9999997327969251], [2, 3.979090758630589, 0.13675758414167993], [16, 21.346364236257617, 0.16557767933875145], [16, 21.87562551229072, 0.1472530084755297], [40, 21.445350078938674, 0.9928153627256894], [8, 12.642109094561857, 0.12477119469827436], [64, 8069.0, 0.0], [16, 23.660196523911065, 0.09719334311698913], [4, 16.929818667528195, 0.001994587277967308], [4, 15.257701399909378, 0.004195514172212866], [10, 12.780313848683722, 0.23621543026603953], [2, 4.206509895247947, 0.12205848698313337], [16, 11.408404447909351, 0.7836029365000337], [4, 5682.0, 0.0]]]
[[1, 1, 1, 1, 0, 1], [1, 1, 1, 0, 0, 1], [1, 1, 1, 0, 0, 1], [0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 1, 0], [1, 1, 0, 0, 0, 1]]

CODING:

```
# plot the heatmap

sns.heatmap(chisqr_result,

        xticklabels=categorical,

        yticklabels=categorical,annot=True)
```


ibm corporate employee attrition analysis.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Connect
Editing
plot the heatmap
sns.heatmap(chisqr_result,
xticklabels=categorical,
yticklabels=categorical,annot=True)
<matplotlib.axes._subplots.AxesSubplot at 0x7fc9820e9128>
BusinessTravel
Department
EducationField
Gender
JobRole
MaritalStatus
1.0
0.8
0.6
0.4
0.2
0.0
Train-Test Split !
[ ] from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import TimeSeriesSplit

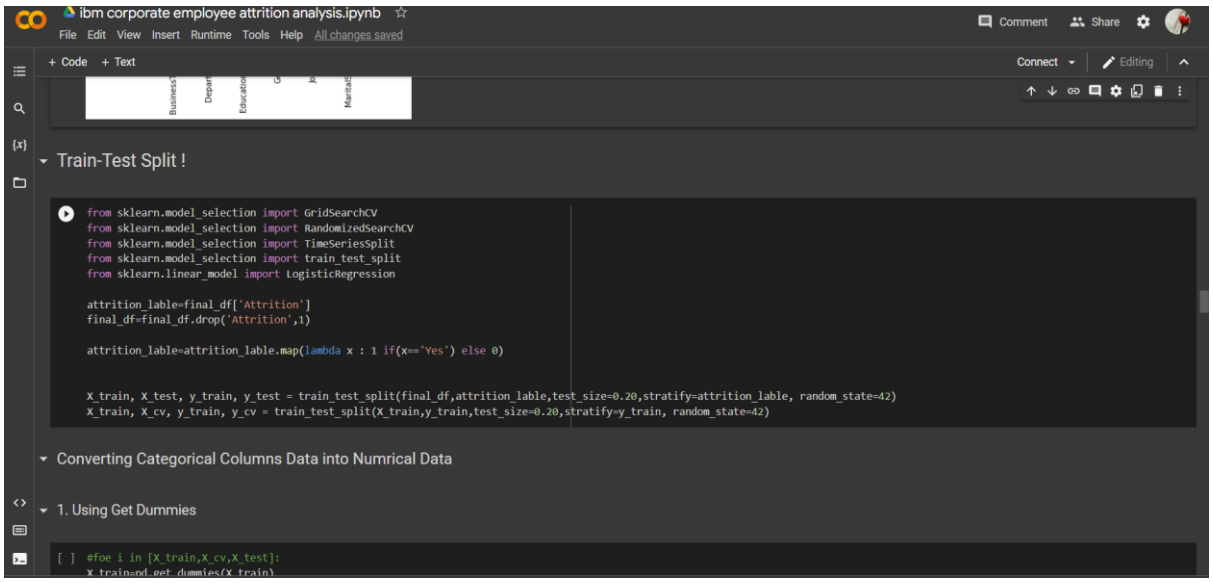
OUTPUT:


BusinessTravel
Department
EducationField
Gender
JobRole
MaritalStatus
1.0
0.8
0.6
0.4
0.2
0.0

TRAIN-TEST SPLIT

CODING:

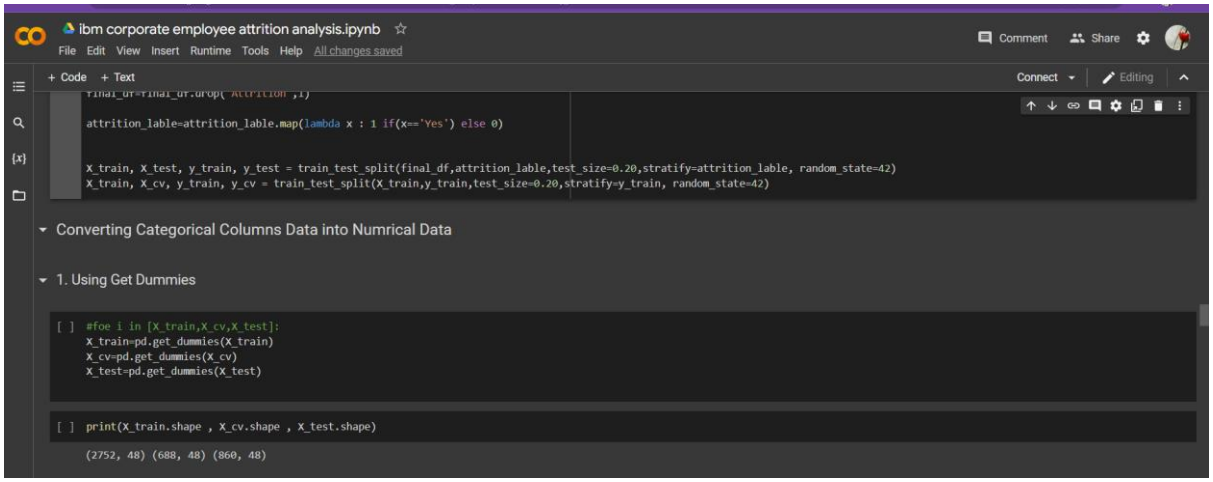
```
from sklearn.model_selection import GridSearchCV

from sklearn.model_selection import RandomizedSearchCV

from sklearn.model_selection import TimeSeriesSplit

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression


attrition_lable=final_df['Attrition']

final_df=final_df.drop('Attrition',1)


attrition_lable=attrition_lable.map(lambda x : 1 if(x=='Yes') else 0)


X_train, X_test, y_train, y_test = train_test_split(final_df,attrition_lable,test_size=0.20,stratify=attrition_lable, random_state=42)

X_train, X_cv, y_train, y_cv = train_test_split(X_train,y_train,test_size=0.20,stratify=y_train, random_state=42)
```



CONVERTING CATEGORICAL COLUMNS DATA INTO NUMERICAL DATA

USING GET DUMMIES

CODING:



#foe i in [X_train,X_cv,X_test]:

X_train=pd.get_dummies(X_train)

X_cv=pd.get_dummies(X_cv)

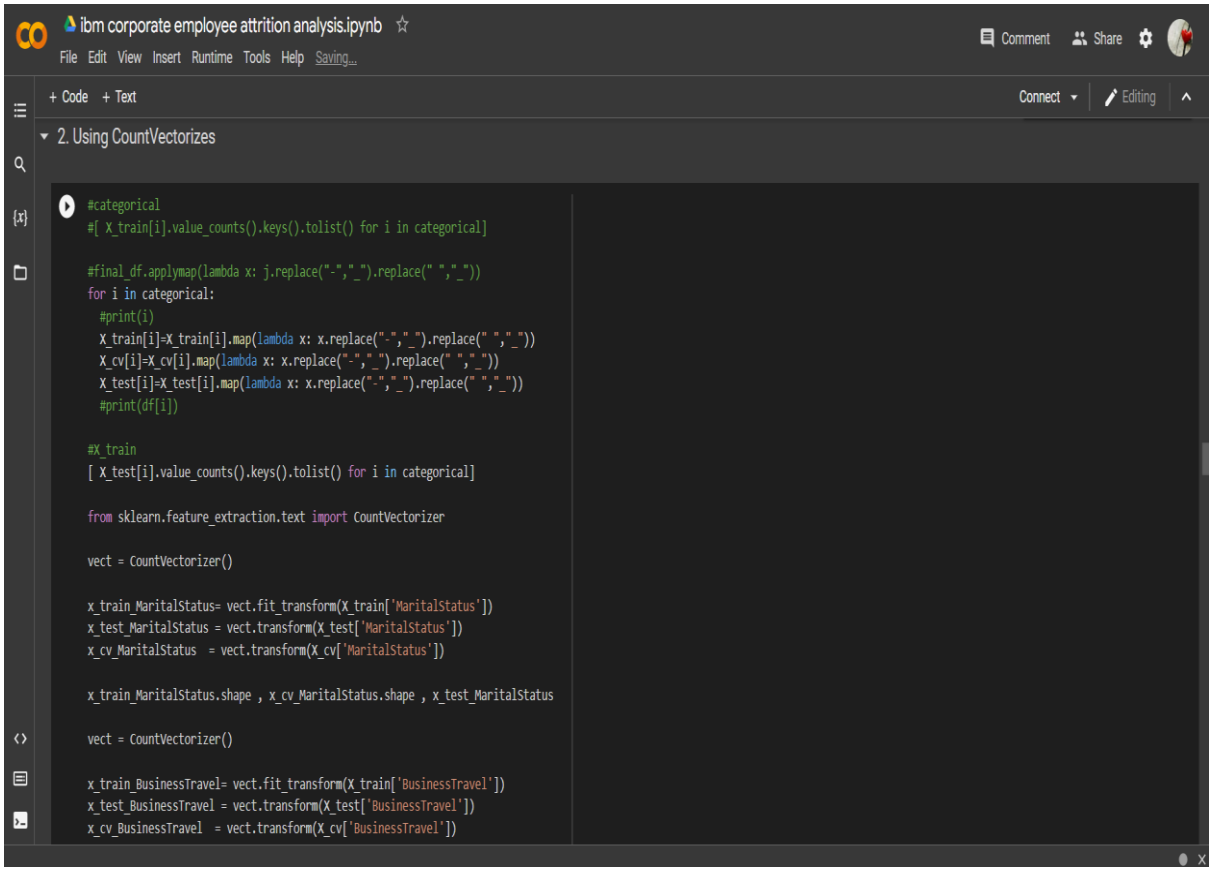
X_test=pd.get_dummies(X_test)

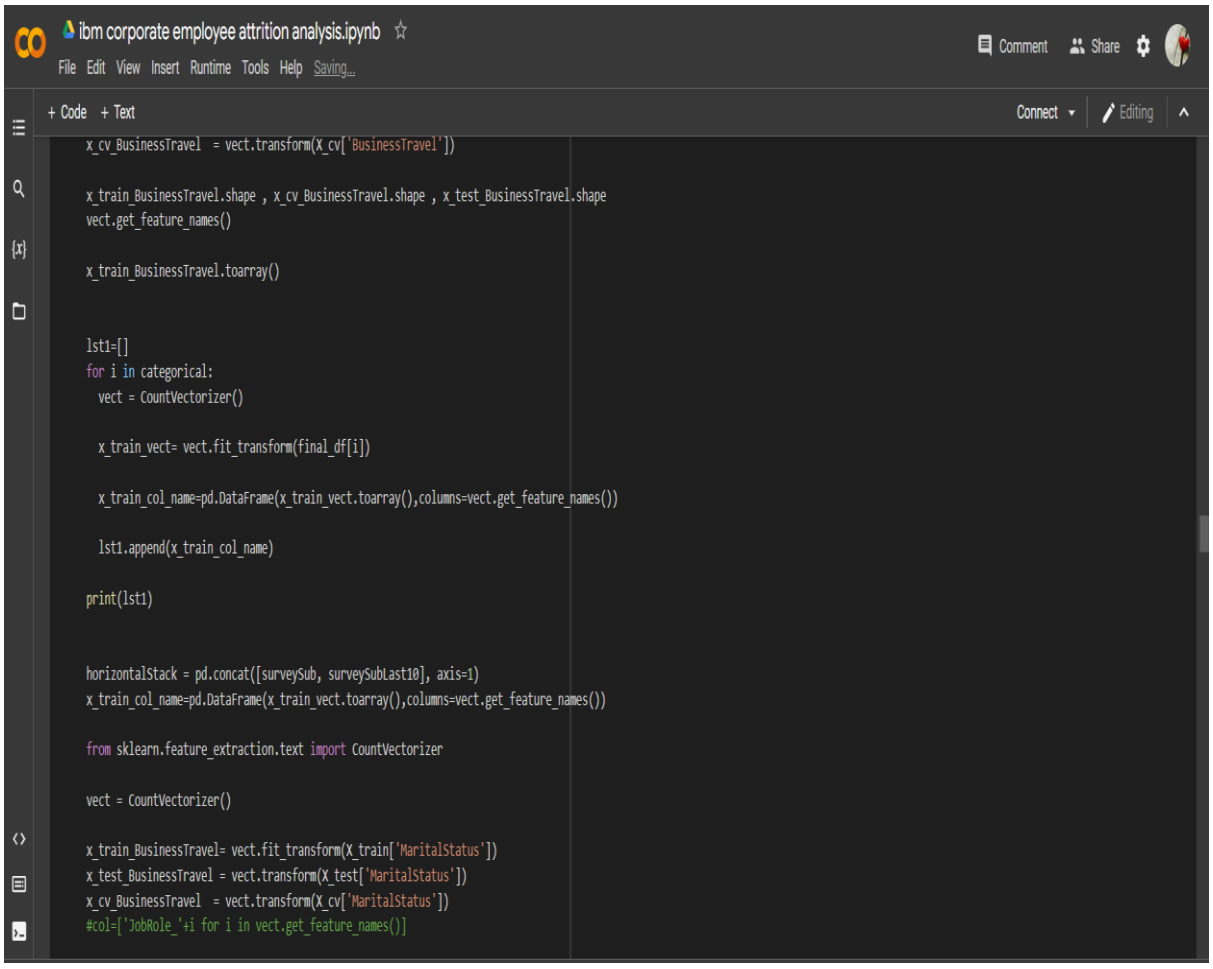
print(X_train.shape , X_cv.shape , X_test.shape)

OUTPUT:

```
(2752, 48) (688, 48) (860, 48)
```

USING COUNT VECTORIZES


ibm corporate employee attrition analysis.ipynb
File Edit View Insert Runtime Tools Help Saving...
Comment
Share
+ Code + Text
Connect
Editing
2. Using CountVectorizes
#categorical
#[X_train[i].value_counts().keys().tolist() for i in categorical]
#final_df.applymap(lambda x: j.replace("-","_").replace(" ","_"))
for i in categorical:
#print(i)
X_train[i]=X_train[i].map(lambda x: x.replace("-","_").replace(" ","_"))
X_cv[i]=X_cv[i].map(lambda x: x.replace("-","_").replace(" ","_"))
X_test[i]=X_test[i].map(lambda x: x.replace("-","_").replace(" ","_"))
#print(df[i])
#X_train
[X_test[i].value_counts().keys().tolist() for i in categorical]
from sklearn.feature_extraction.text import CountVectorizer
vect = CountVectorizer()
x_train_MaritalStatus= vect.fit_transform(X_train['MaritalStatus'])
x_test_MaritalStatus = vect.transform(X_test['MaritalStatus'])
x_cv_MaritalStatus = vect.transform(X_cv['MaritalStatus'])
x_train_MaritalStatus.shape , x_cv_MaritalStatus.shape , x_test_MaritalStatus
vect = CountVectorizer()
x_train_BusinessTravel= vect.fit_transform(X_train['BusinessTravel'])
x_test_BusinessTravel = vect.transform(X_test['BusinessTravel'])
x_cv_BusinessTravel = vect.transform(X_cv['BusinessTravel'])


ibm corporate employee attrition analysis.ipynb
File Edit View Insert Runtime Tools Help Saving...
Comment
Share
+ Code + Text
Connect
Editing
x_cv_BusinessTravel = vect.transform(X_cv['BusinessTravel'])
x_train_BusinessTravel.shape , x_cv_BusinessTravel.shape , x_test_BusinessTravel.shape
vect.get_feature_names()
x_train_BusinessTravel.toarray()
lst1=[]
for i in categorical:
vect = CountVectorizer()
x_train_vect= vect.fit_transform(final_df[i])
x_train_col_name=pd.DataFrame(x_train_vect.toarray(),columns=vect.get_feature_names())
lst1.append(x_train_col_name)
print(lst1)
horizontalStack = pd.concat([surveySub, surveySubLast10], axis=1)
x_train_col_name=pd.DataFrame(x_train_vect.toarray(),columns=vect.get_feature_names())
from sklearn.feature_extraction.text import CountVectorizer
vect = CountVectorizer()
x_train_BusinessTravel= vect.fit_transform(X_train['MaritalStatus'])
x_test_BusinessTravel = vect.transform(X_test['MaritalStatus'])
x_cv_BusinessTravel = vect.transform(X_cv['MaritalStatus'])
#col=['JobRole_'+i for i in vect.get_feature_names()]

CODING:

```
#categorical
#[ X_train[i].value_counts().keys().tolist() for i in categorical]


#final_df.applymap(lambda x: j.replace("-","_").replace(" ","_"))
for i in categorical:
  #print(i)
  X_train[i]=X_train[i].map(lambda x: x.replace("-","_").replace(" ","_"))
  X_cv[i]=X_cv[i].map(lambda x: x.replace("-","_").replace(" ","_"))
  X_test[i]=X_test[i].map(lambda x: x.replace("-","_").replace(" ","_"))
  #print(df[i])


#X_train
[ X_test[i].value_counts().keys().tolist() for i in categorical]


from sklearn.feature_extraction.text import CountVectorizer


vect = CountVectorizer()


x_train_MaritalStatus= vect.fit_transform(X_train['MaritalStatus'])
x_test_MaritalStatus = vect.transform(X_test['MaritalStatus'])
x_cv_MaritalStatus  = vect.transform(X_cv['MaritalStatus'])


x_train_MaritalStatus.shape , x_cv_MaritalStatus.shape , x_test_MaritalStatus


vect = CountVectorizer()
```

```
x_train_BusinessTravel= vect.fit_transform(X_train['BusinessTravel'])

x_test_BusinessTravel = vect.transform(X_test['BusinessTravel'])

x_cv_BusinessTravel  = vect.transform(X_cv['BusinessTravel'])


x_train_BusinessTravel.shape , x_cv_BusinessTravel.shape , x_test_BusinessTravel.shape

vect.get_feature_names()


x_train_BusinessTravel.toarray()


lst1=[]

for i in categorical:

  vect = CountVectorizer()


  x_train_vect= vect.fit_transform(final_df[i])


  x_train_col_name=pd.DataFrame(x_train_vect.toarray(),columns=vect.get_feature_names())


  lst1.append(x_train_col_name)


print(lst1)


horizontalStack = pd.concat([surveySub, surveySubLast10], axis=1)

x_train_col_name=pd.DataFrame(x_train_vect.toarray(),columns=vect.get_feature_names())
```

```
from sklearn.feature_extraction.text import CountVectorizer

vect = CountVectorizer()

x_train_BusinessTravel= vect.fit_transform(X_train['MaritalStatus'])
x_test_BusinessTravel = vect.transform(X_test['MaritalStatus'])
x_cv_BusinessTravel  = vect.transform(X_cv['MaritalStatus'])
#col=['JobRole_'+i for i in vect.get_feature_names()]

pd.DataFrame(x_train_BusinessTravel.toarray(),columns=col)

final_df.columns.tolist()
```

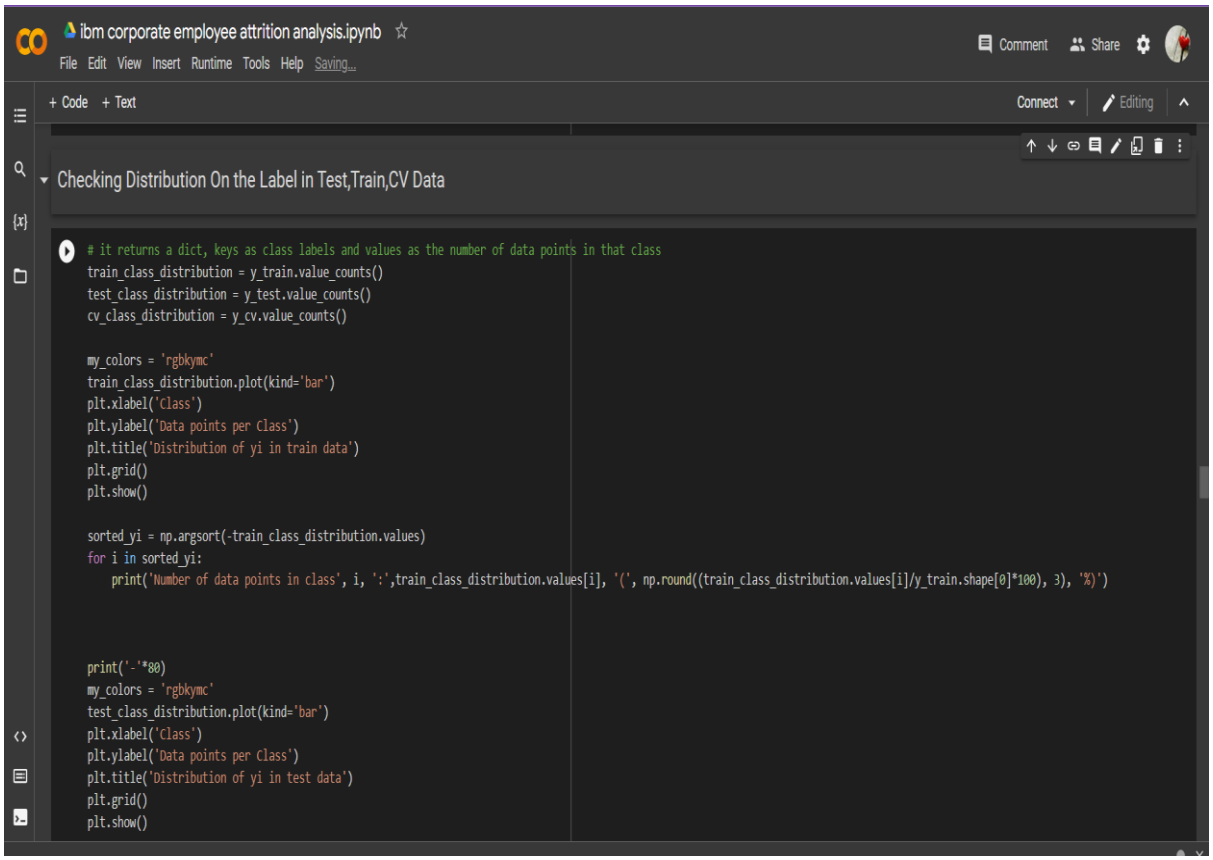
CHECKING DISTRIBUTION ON THE LABEL IN TEST, TRAIN, CV DATA

CODING:

```
# it returns a dict, keys as class labels and values as the number of data points in
 that class
train_class_distribution = y_train.value_counts()
test_class_distribution = y_test.value_counts()
cv_class_distribution = y_cv.value_counts()

my_colors = 'rgbkymc'
train_class_distribution.plot(kind='bar')
```

```
plt.xlabel('Class')
plt.ylabel('Data points per Class')
plt.title('Distribution of yi in train data')
plt.grid()
plt.show()

sorted_yi = np.argsort(-train_class_distribution.values)
for i in sorted_yi:
    print('Number of data points in class', i, ':',train_class_distribution.values[i], '('
, np.round((train_class_distribution.values[i]/y_train.shape[0]*100), 3), '%)')



print('-'*80)
my_colors = 'rgbkymc'
test_class_distribution.plot(kind='bar')
plt.xlabel('Class')
plt.ylabel('Data points per Class')
plt.title('Distribution of yi in test data')
plt.grid()
plt.show()

sorted_yi = np.argsort(-test_class_distribution.values)
for i in sorted_yi:
    print('Number of data points in class', i+1, ':',test_class_distribution.values[i], '
(', np.round((test_class_distribution.values[i]/y_test.shape[0]*100), 3), '%)')
```

```
print('-'*80)
my_colors = 'rgbkymc'
cv_class_distribution.plot(kind='bar')
plt.xlabel('Class')
plt.ylabel('Data points per Class')
plt.title('Distribution of yi in cross validation data')
plt.grid()
plt.show()

sorted_yi = np.argsort(-train_class_distribution.values)
for i in sorted_yi:
    print('Number of data points in class', i+1, ':',cv_class_distribution.values[i], '('
, np.round((cv_class_distribution.values[i]/y_cv.shape[0]*100), 3), '%)')
```



ibm corporate employee attrition analysis.ipynb ☆

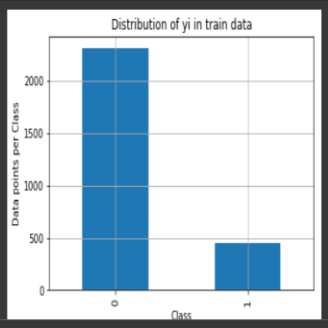
File Edit View Insert Runtime Tools Help Saving...

Comment Share

+ Code + Text

Connect Editing

```
plt.show()

sorted_yi = np.argsort(-test_class_distribution.values)
for i in sorted_yi:
    print('Number of data points in class', i+1, ':',test_class_distribution.values[i], '(', np.round((test_class_distribution.values[i]/y_test.shape[0]*100), 3), '%)')

print('-'*80)
my_colors = 'rgbkymc'
cv_class_distribution.plot(kind='bar')
plt.xlabel('Class')
plt.ylabel('Data points per Class')
plt.title('Distribution of yi in cross validation data')
plt.grid()
plt.show()

sorted_yi = np.argsort(-train_class_distribution.values)
for i in sorted_yi:
    print('Number of data points in class', i+1, ':',cv_class_distribution.values[i], '(', np.round((cv_class_distribution.values[i]/y_cv.shape[0]*100), 3), '%)')
```



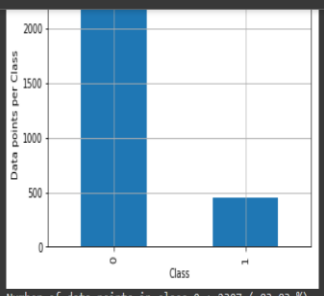
ibm corporate employee attrition analysis.ipynb ☆

File Edit View Insert Runtime Tools Help Saving...

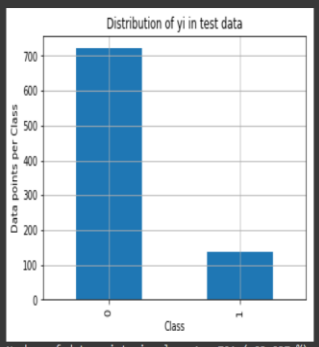
Comment Share

+ Code + Text

Connect Editing

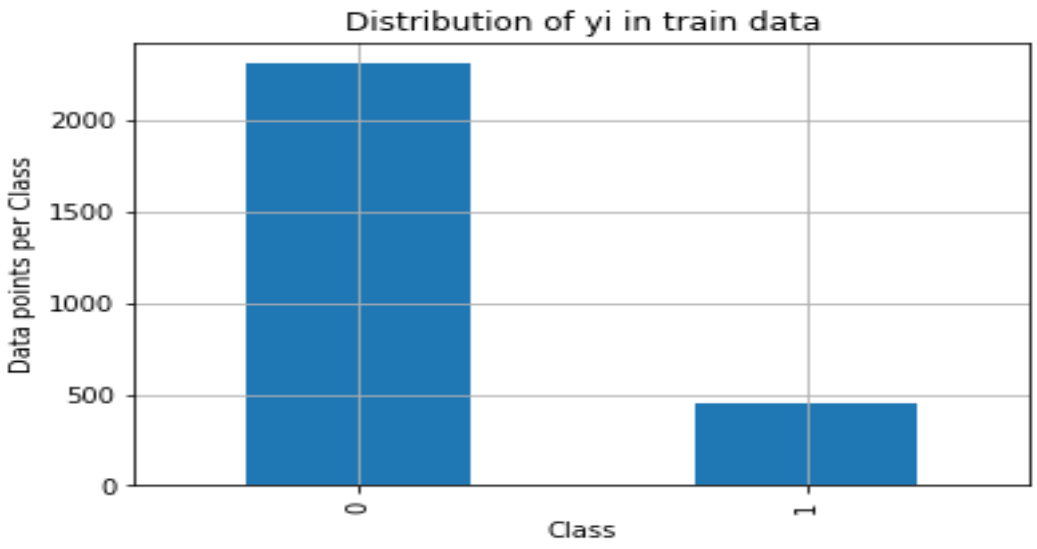



```
Number of data points in class 0 : 2307 ( 83.83 %)
Number of data points in class 1 : 445 ( 16.17 %)
--------------------------------------------------------------------------------
```



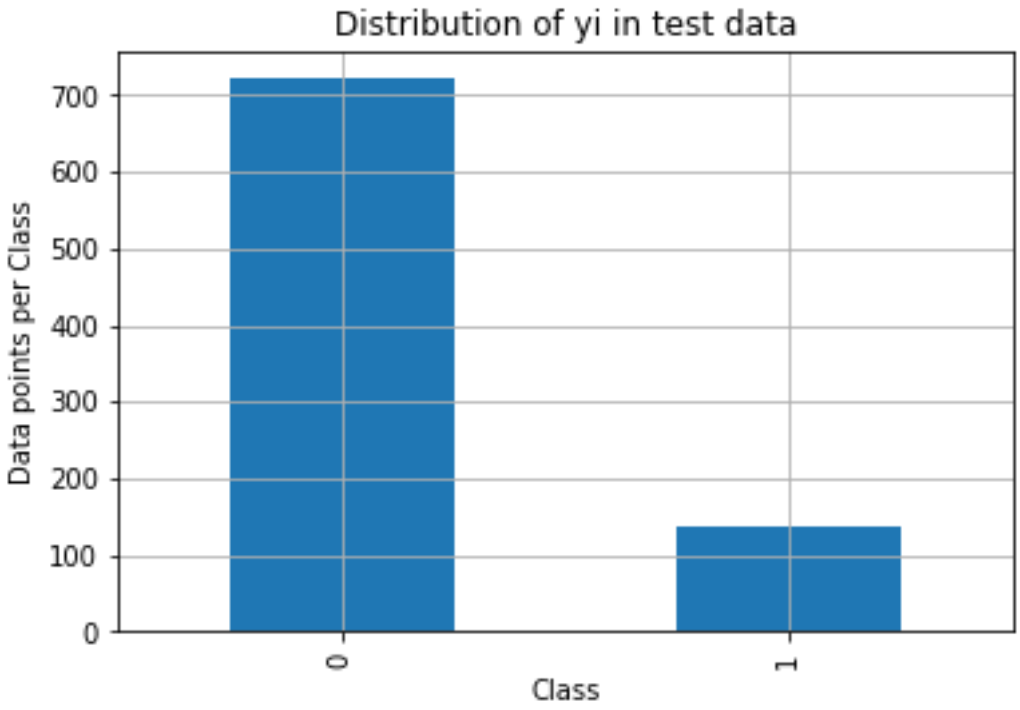

```
Number of data points in class 1 : 721 ( 83.837 %)
Number of data points in class 2 : 139 ( 16.163 %)
--------------------------------------------------------------------------------
```

OUTPUT:



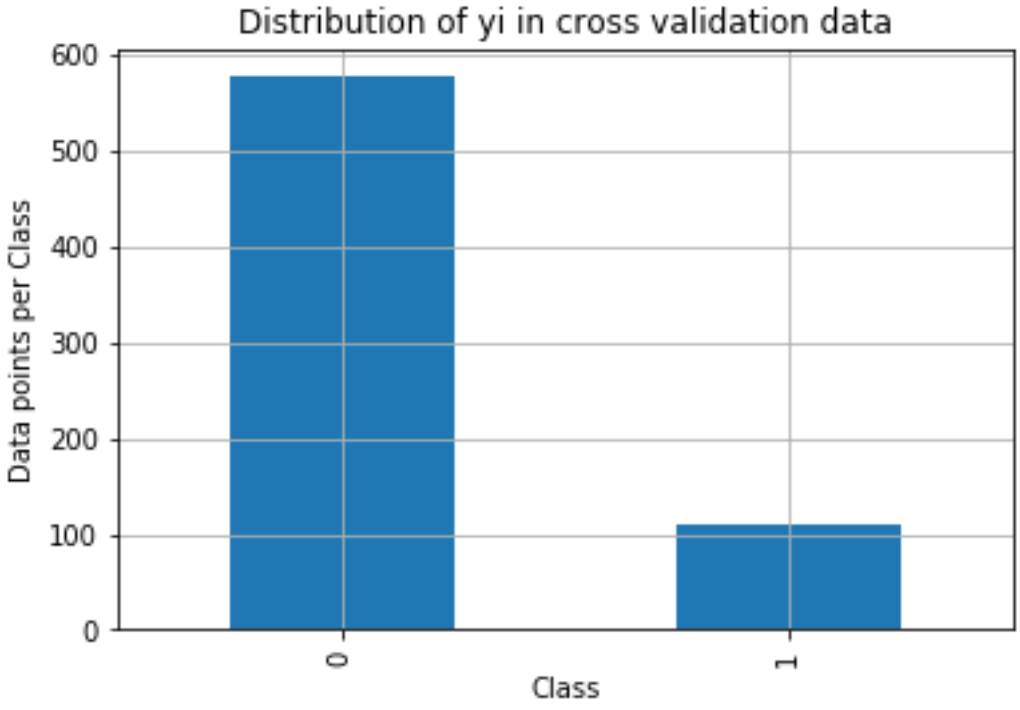
Number of data points in class 0 : 2307 (83.83 %)

Number of data points in class 1 : 445 (16.17 %)



Number of data points in class 1 : 721 (83.837 %)

Number of data points in class 2 : 139 (16.163 %)



Number of data points in class 1 : 577 (83.866 %)

Number of data points in class 2 : 111 (16.134 %)