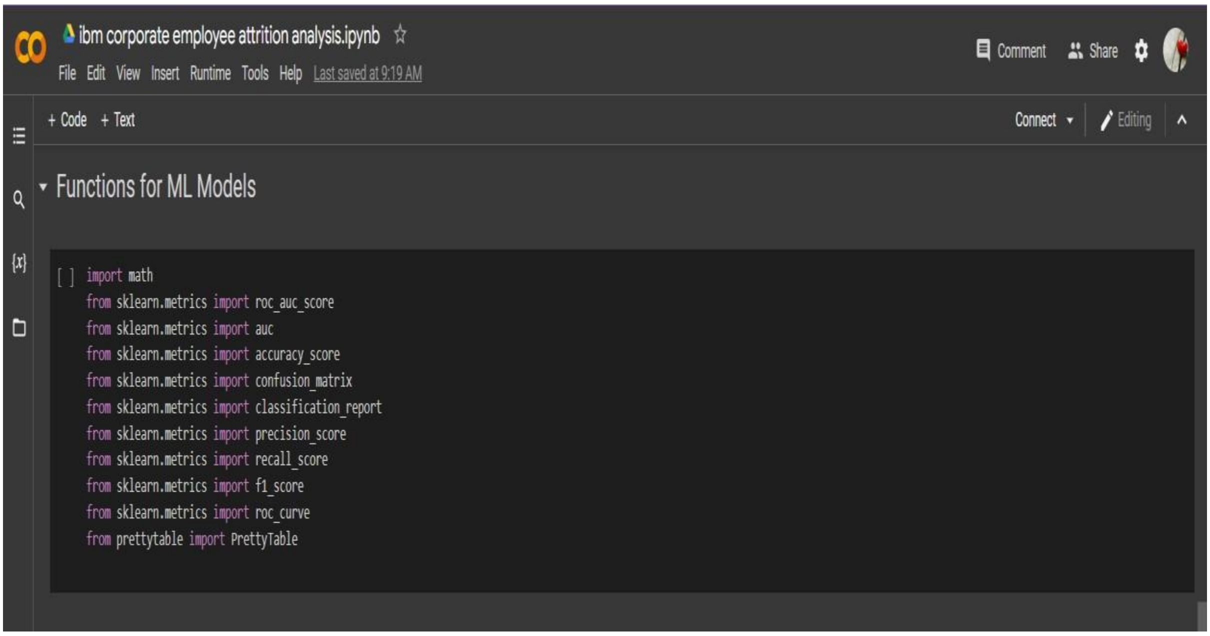
Project Development

Delivery Of Sprint-4

Date	03 October 2022
Team ID	PNT2022TMID06047
Project Name	Project - Corporate Employee Attrition Analytics

MODEL BUILDING, MODEL EVALUATION, VISUALIZATION CHARTS AND DASHBOARD CREATION

FUNCTIONS FOR ML MODELS



CODING:

```
import math

from sklearn.metrics import roc_auc_score

from sklearn.metrics import auc
```

from sklearn.metrics import accuracy_score

from sklearn.metrics import confusion_matrix

from sklearn.metrics import

classification_report from sklearn.metrics

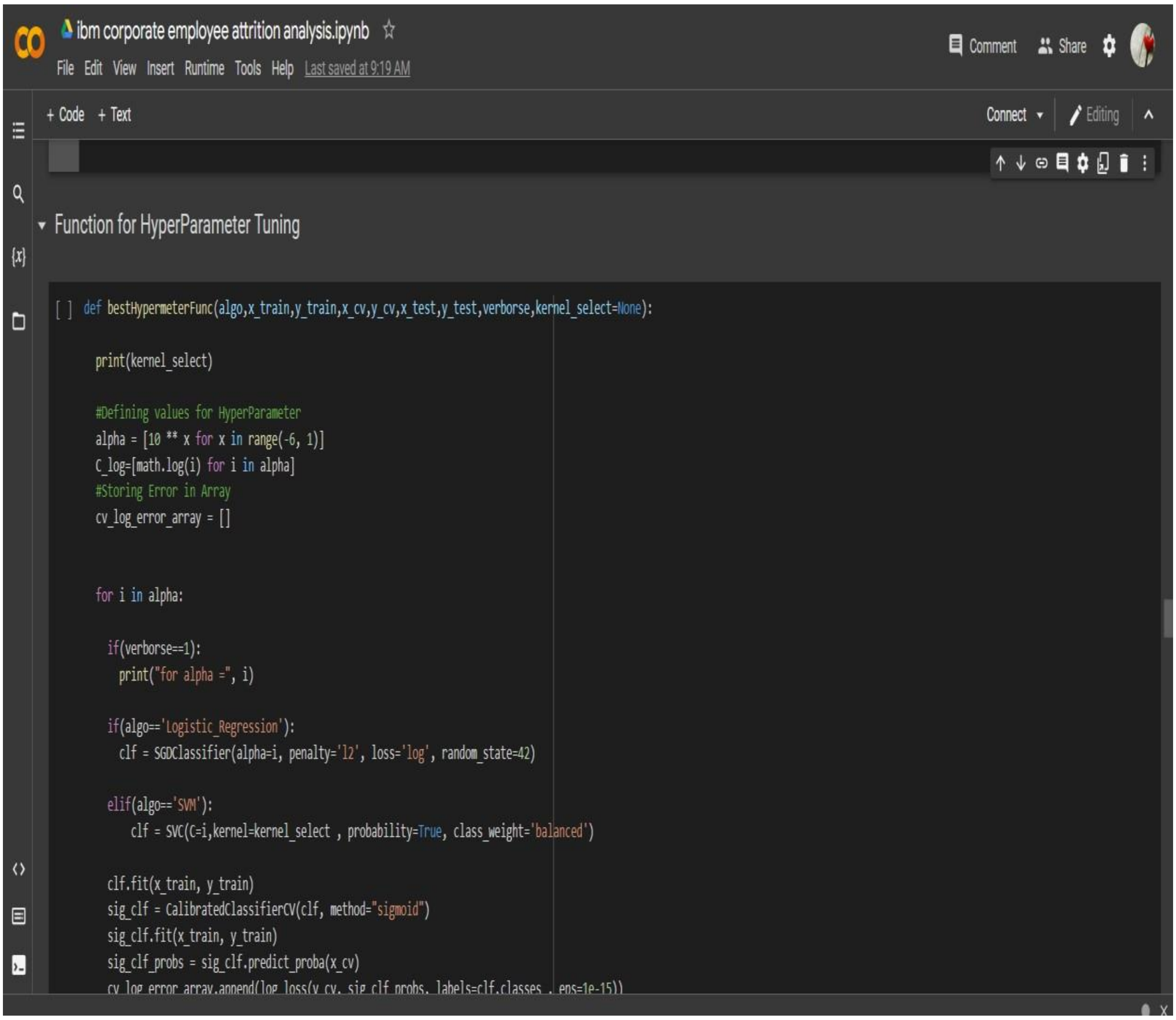
import precision_score from sklearn.metrics

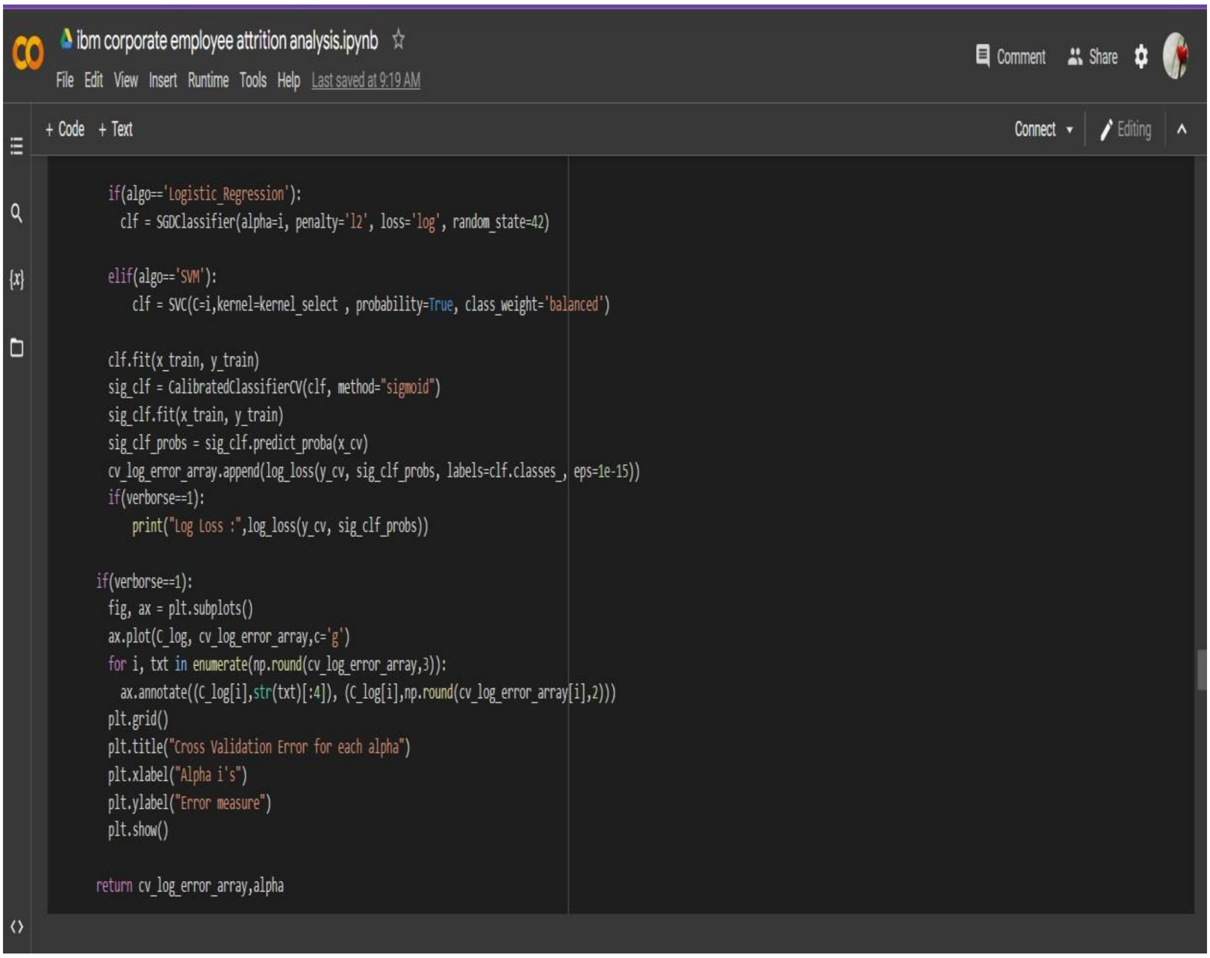
import recall_score from sklearn.metrics import

f1_score from sklearn.metrics import roc_curve

from prettytable import PrettyTable

FUNCTION FOR HYPER PARAMETER TUNING





```
    if(algo=='Logistic Regression'):
      clf = SGDClassifier(alpha=i, penalty='l2', loss='log', random_state=42)

    elif(algo=='SVM'):
        clf = SVC(C=i,kernel=kernel_select , probability=True, class_weight='balanced')

    clf.fit(x_train, y_train)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(x_train, y_train)
    sig_clf_probs = sig_clf.predict_proba(x_cv)
    cv_log_error_array.append(log_loss(y_cv, sig_clf_probs, labels=clf.classes_, eps=1e-15))
    if(verborse==1):
        print("Log Loss :",log_loss(y_cv, sig_clf_probs))

  if(verborse==1):
    fig, ax = plt.subplots()
    ax.plot(C_log, cv_log_error_array,c='g')
    for i, txt in enumerate(np.round(cv_log_error_array,3)):
      ax.annotate((C_log[i],str(txt)[:4]), (C_log[i],np.round(cv_log_error_array[i],2)))
    plt.grid()
    plt.title("Cross Validation Error for each alpha")
    plt.xlabel("Alpha i's")
    plt.ylabel("Error measure")
    plt.show()

  return cv_log_error_array,alpha
```

CODING:

def bestHypermeterFunc(algo,x_train,y_train,x_cv,y_cv,x_test,y_test,verborse,kernel_select=None):

print(kernel_select)

#Defining values for HyperParameter

alpha = [10 ** x for x in range(-6, 1)]

C_log=[math.log(i) for i in alpha]

#Storing Error in Array

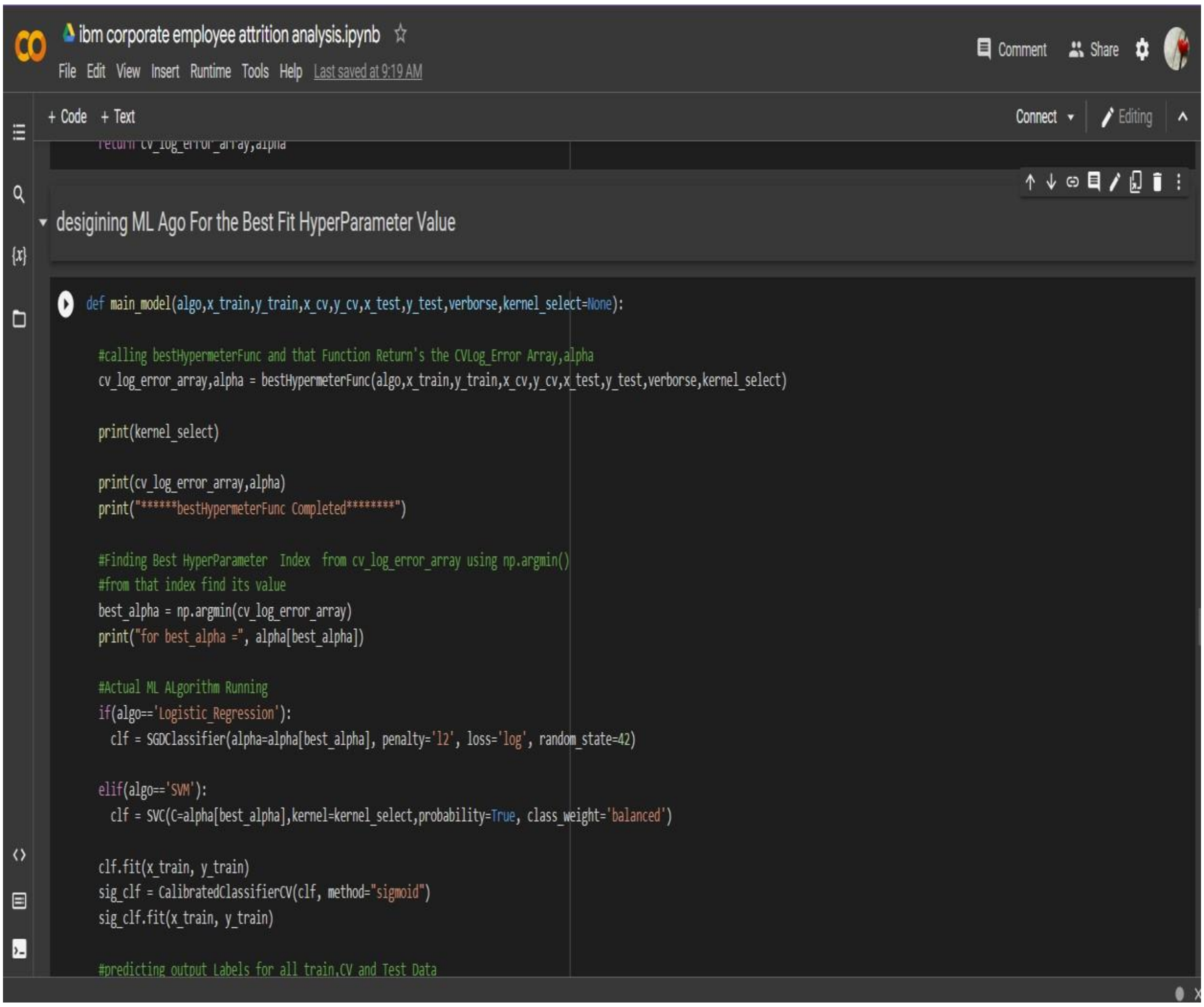
cv_log_error_array = []   for i in alpha:

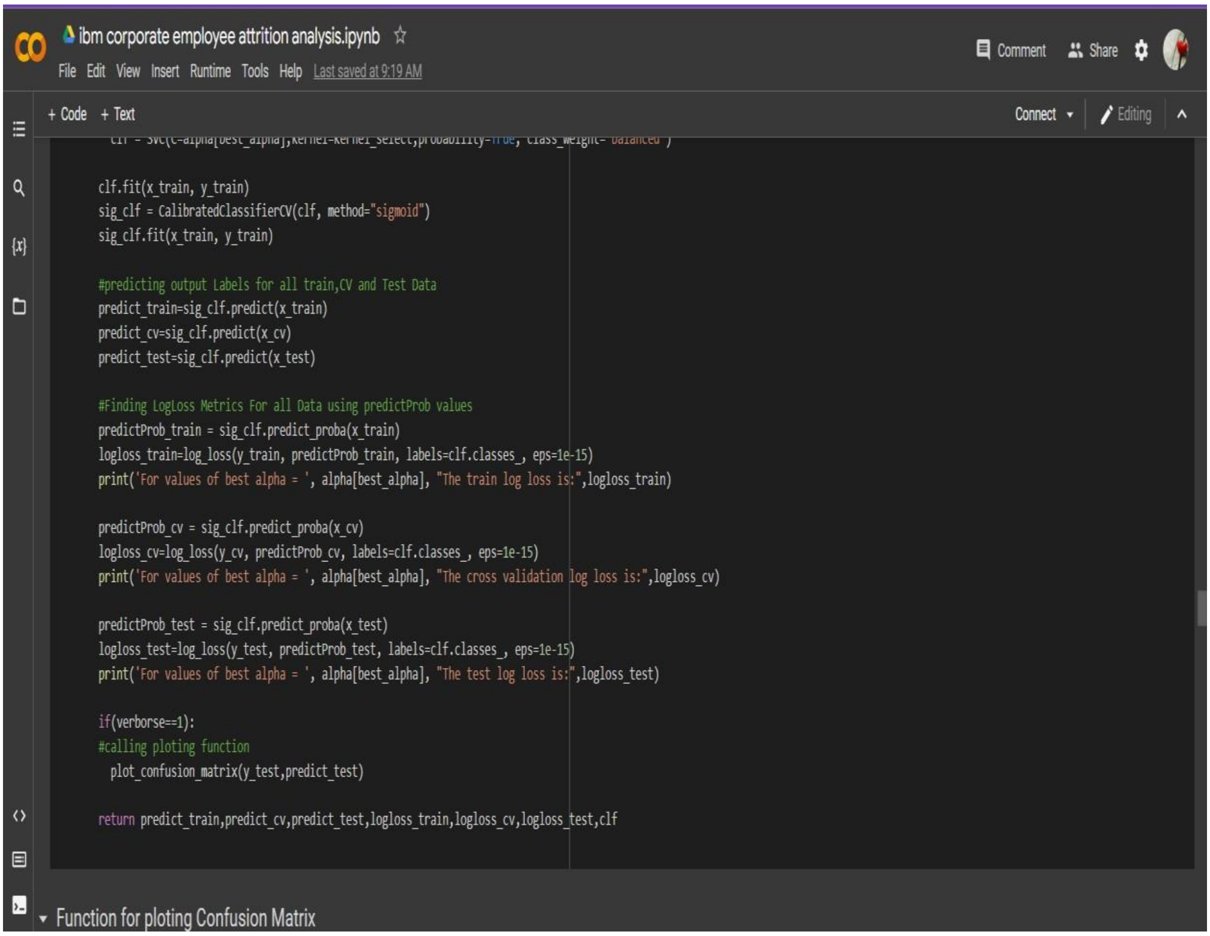
```
    if(verborse==1):
      print("for alpha =", i)

    if(algo=='Logistic_Regression'):
      clf = SGDClassifier(alpha=i, penalty='l2', loss='log', random_state=42)

    elif(algo=='SVM'):
        clf = SVC(C=i,kernel=kernel_select , probability=True, class_weight='balance
d')

    clf.fit(x_train, y_train)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(x_train, y_train)
    sig_clf_probs = sig_clf.predict_proba(x_cv)
    cv_log_error_array.append(log_loss(y_cv,  sig_clf_probs,  labels=clf.classes_,
e ps=1e-15))     if(verborse==1):
        print("Log Loss :",log_loss(y_cv, sig_clf_probs))

  if(verborse==1):
    fig, ax = plt.subplots()
    ax.plot(C_log, cv_log_error_array,c='g')     for i, txt in
enumerate(np.round(cv_log_error_array,3)):
ax.annotate((C_log[i],str(txt)[:4]),
(C_log[i],np.round(cv_log_error_array[i],2)
))
plt.grid()
```

plt.title("Cross Validation Error for each alpha") plt.xlabel("Alpha i's") plt.ylabel("Error measure") plt.show()

return cv_log_error_array,alpha

DESIGNING ML AGO FOR THE BEST FIT HYPER PARAMETER VALUE





CODING:

def main_model(algo,x_train,y_train,x_cv,y_cv,x_test,y_test,verborse,kernel_se lect=None):

#calling bestHypermeterFunc and that Function Return's the CVLog_Error Arra y,alpha

cv_log_error_array,alpha = bestHypermeterFunc(algo,x_train,y_train,x_cv,y_cv ,x_test,y_test,verborse,kernel_select)

print(kernel_select)

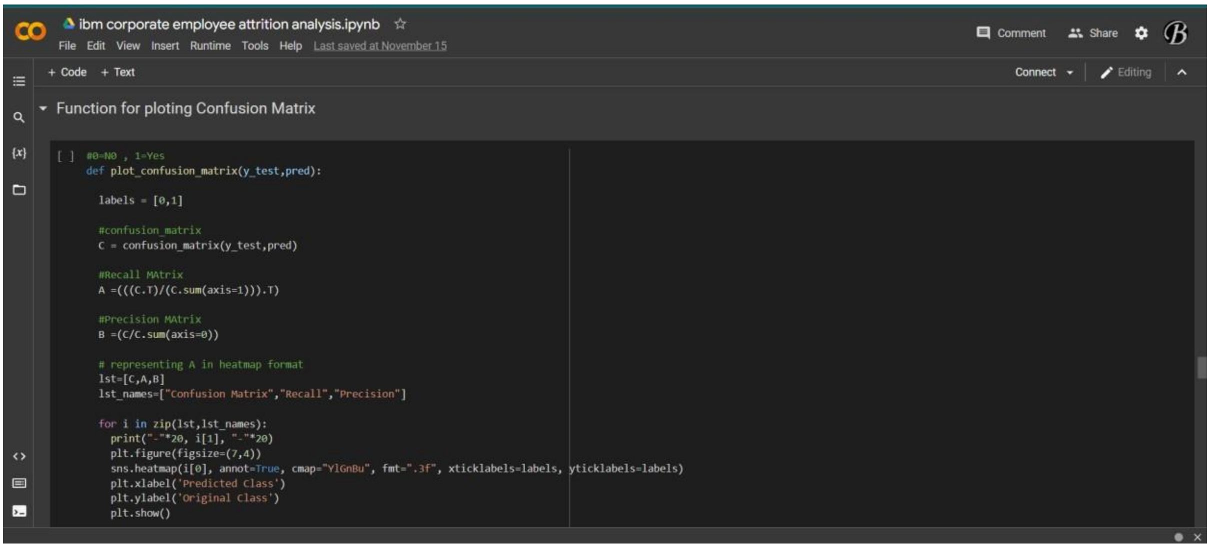
print(cv_log_error_array,alpha)

print("******bestHypermeterFunc Completed********")

```
 #Finding Best HyperParameter  Index  from cv_log_error_array using np.argmi
n()
 #from that index find its value   best_alpha
= np.argmin(cv_log_error_array)   print("for
best_alpha =", alpha[best_alpha])

 #Actual ML ALgorithm Running
if(algo=='Logistic_Regression'):
   clf = SGDClassifier(alpha=alpha[best_alpha], penalty='l2', loss='log', random_s
tate=42)

 elif(algo=='SVM'):
   clf = SVC(C=alpha[best_alpha],kernel=kernel_select,probability=True, class_w
eight='balanced')

 clf.fit(x_train, y_train)
 sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(x_train, y_train)

 #predicting output Labels for all train,CV and Test Data
predict_train=sig_clf.predict(x_train)
predict_cv=sig_clf.predict(x_cv)
predict_test=sig_clf.predict(x_test)

  #Finding LogLoss Metrics For all Data using predictProb values
  predictProb_train = sig_clf.predict_proba(x_train)
```

```
  logloss_train=log_loss(y_train, predictProb_train, labels=clf.classes_, eps=1e-
15)
  print('For values of best alpha = ', alpha[best_alpha], "The train log loss is:",logl
oss_train)

  predictProb_cv = sig_clf.predict_proba(x_cv)
  logloss_cv=log_loss(y_cv,    predictProb_cv,    labels=clf.classes_,    eps=1e-15)
print('For values of best alpha = ', alpha[best_alpha], "The cross validation log l
oss is:",logloss_cv)

  predictProb_test = sig_clf.predict_proba(x_test)
  logloss_test=log_loss(y_test, predictProb_test, labels=clf.classes_, eps=1e-15)
print('For values of best alpha = ', alpha[best_alpha], "The test log loss is:",loglo
ss_test)

  if(verborse==1):
#calling ploting function
    plot_confusion_matrix(y_test,predict_test)

  return
predict_train,predict_cv,predict_test,logloss_train,logloss_cv,logloss_te st,clf
```

FUNCTION FOR PLOTTING CONFUSION MATRIX

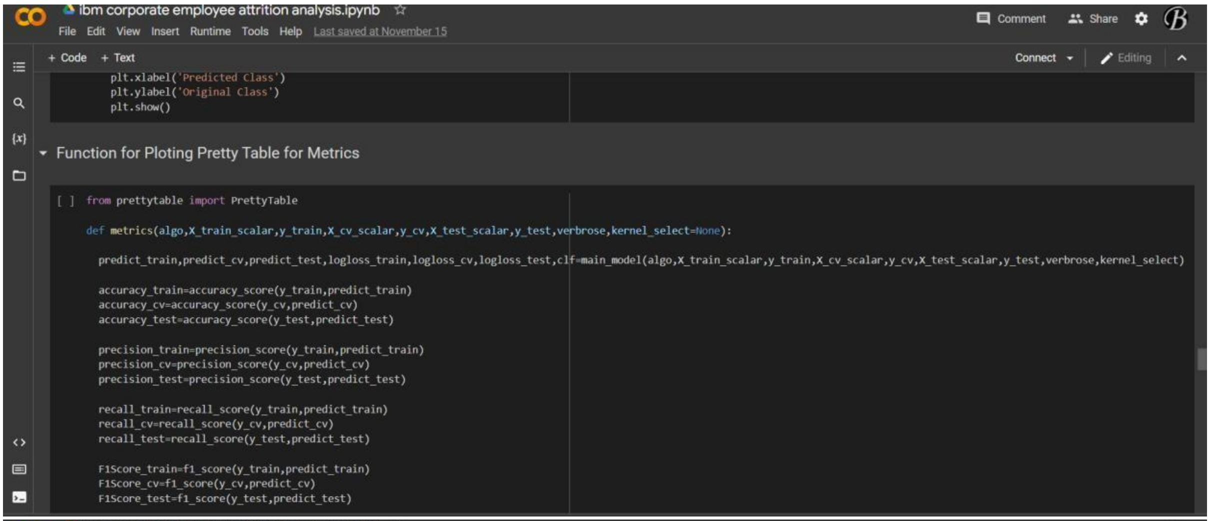


CODING:

```
#0=N0 , 1=Yes def

plot_confusion_matrix(y_test,pred):


  labels = [0,1]


  #confusion_matrix

  C = confusion_matrix(y_test,pred)


  #Recall MAtrix

A =(((C.T)/(C.sum(axis=1))).T)


  #Precision MAtrix

B =(C/C.sum(axis=0))


  # representing A in heatmap format

  lst=[C,A,B]
```

```
lst_names=["Confusion Matrix","Recall","Precision"]

for i in zip(lst,lst_names):     print("-
"*20, i[1], "-"*20)
plt.figure(figsize=(7,4))
   sns.heatmap(i[0], annot=True, cmap="YlGnBu", fmt=".3f", xticklabels=labels,
y ticklabels=labels)      plt.xlabel('Predicted Class')      plt.ylabel('Original Class')
plt.show()
```

FUNCTION FOR PLOTTING PRETTY TABLE FOR METRICS

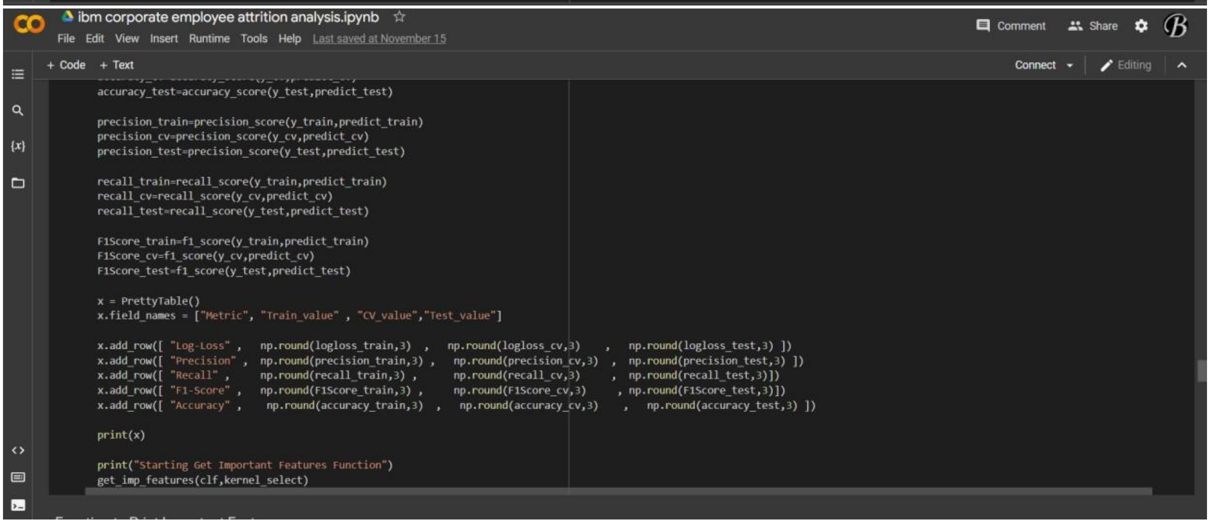


ibm corporate employee attrition analysis.ipynb
File Edit View Insert Runtime Tools Help Last saved at November 15
Comment Share
+ Code + Text Connect Editing

```
        plt.xlabel('Predicted Class')
        plt.ylabel('Original Class')
        plt.show()
```

Function for Ploting Pretty Table for Metrics

```
[ ] from prettytable import PrettyTable

    def metrics(algo,X_train_scalar,y_train,X_cv_scalar,y_cv,X_test_scalar,y_test,verbrose,kernel_select=None):

      predict_train,predict_cv,predict_test,logloss_train,logloss_cv,logloss_test,clf=main_model(algo,X_train_scalar,y_train,X_cv_scalar,y_cv,X_test_scalar,y_test,verbrose,kernel_select)

      accuracy_train=accuracy_score(y_train,predict_train)
      accuracy_cv=accuracy_score(y_cv,predict_cv)
      accuracy_test=accuracy_score(y_test,predict_test)

      precision_train=precision_score(y_train,predict_train)
      precision_cv=precision_score(y_cv,predict_cv)
      precision_test=precision_score(y_test,predict_test)

      recall_train=recall_score(y_train,predict_train)
      recall_cv=recall_score(y_cv,predict_cv)
      recall_test=recall_score(y_test,predict_test)

      F1Score_train=f1_score(y_train,predict_train)
      F1Score_cv=f1_score(y_cv,predict_cv)
      F1Score_test=f1_score(y_test,predict_test)
```



ibm corporate employee attrition analysis.ipynb
File Edit View Insert Runtime Tools Help Last saved at November 15
Comment Share
+ Code + Text Connect Editing

```
      accuracy_test=accuracy_score(y_test,predict_test)

      precision_train=precision_score(y_train,predict_train)
      precision_cv=precision_score(y_cv,predict_cv)
      precision_test=precision_score(y_test,predict_test)

      recall_train=recall_score(y_train,predict_train)
      recall_cv=recall_score(y_cv,predict_cv)
      recall_test=recall_score(y_test,predict_test)

      F1Score_train=f1_score(y_train,predict_train)
      F1Score_cv=f1_score(y_cv,predict_cv)
      F1Score_test=f1_score(y_test,predict_test)

      x = PrettyTable()
      x.field_names = ["Metric", "Train_value" , "CV_value","Test_value"]

      x.add_row([ "Log-Loss" ,   np.round(logloss_train,3)  ,   np.round(logloss_cv,3)    ,   np.round(logloss_test,3) ])
      x.add_row([ "Precision" ,  np.round(precision_train,3) ,   np.round(precision_cv,3)  ,  np.round(precision_test,3) ])
      x.add_row([ "Recall" ,     np.round(recall_train,3) ,      np.round(recall_cv,3)     ,  np.round(recall_test,3)])
      x.add_row([ "F1-Score" ,   np.round(F1Score_train,3) ,     np.round(F1Score_cv,3)    ,  np.round(F1Score_test,3)])
      x.add_row([ "Accuracy" ,    np.round(accuracy_train,3)  ,   np.round(accuracy_cv,3)    ,   np.round(accuracy_test,3) ])

      print(x)

      print("Starting Get Important Features Function")
      get_imp_features(clf,kernel_select)
```

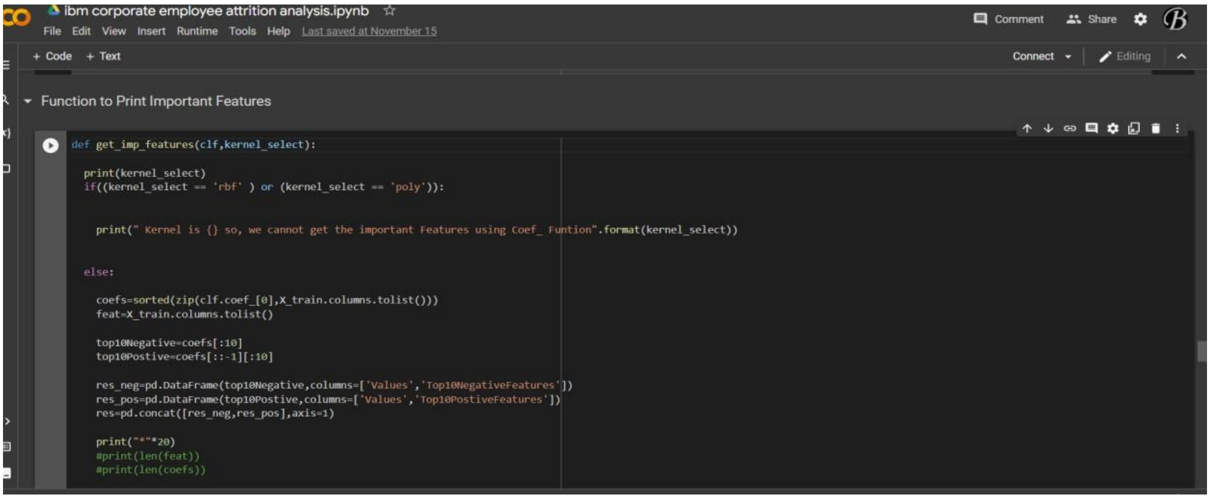
CODING:

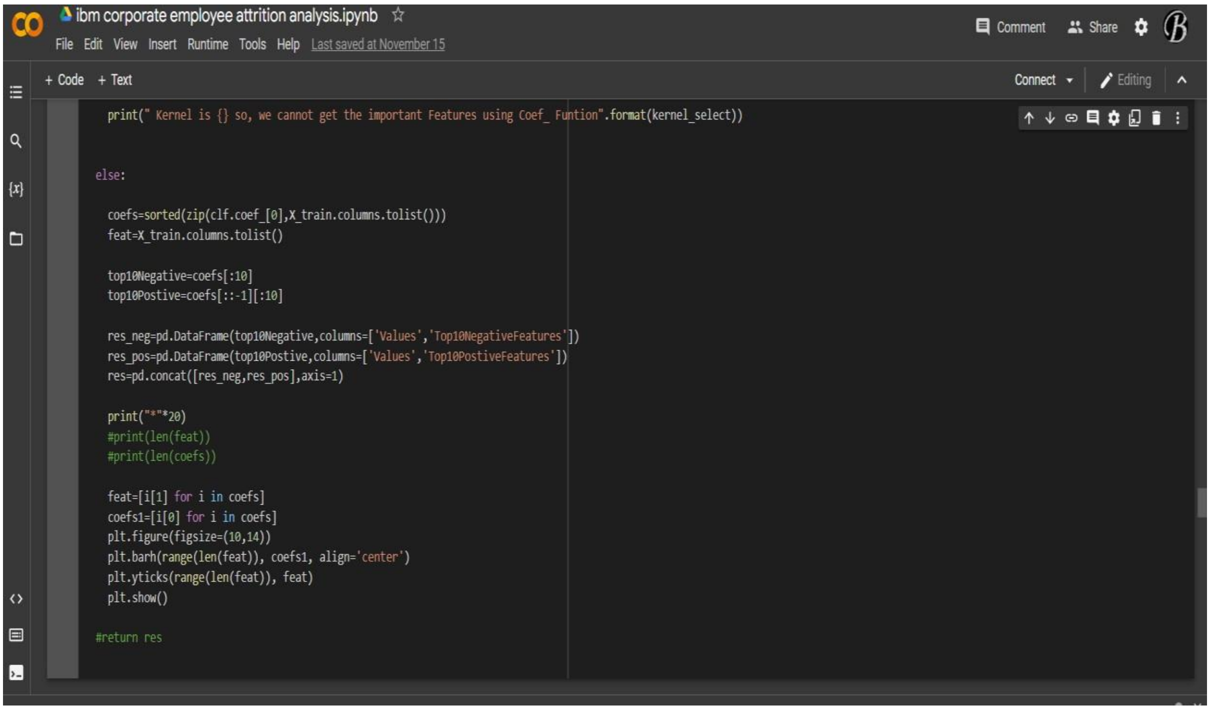
```
from prettytable import PrettyTable
```

```
def metrics(algo,X_train_scalar,y_train,X_cv_scalar,y_cv,X_test_scalar,y_test,ve
rbrose,kernel_select=None):

  predict_train,predict_cv,predict_test,logloss_train,logloss_cv,logloss_test,clf=
main_model(algo,X_train_scalar,y_train,X_cv_scalar,y_cv,X_test_scalar,y_test,v
erbrose,kernel_select)

  accuracy_train=accuracy_score(y_train,predict_train)
accuracy_cv=accuracy_score(y_cv,predict_cv)
accuracy_test=accuracy_score(y_test,predict_test)

  precision_train=precision_score(y_train,predict_train)
precision_cv=precision_score(y_cv,predict_cv)
precision_test=precision_score(y_test,predict_test)

  recall_train=recall_score(y_train,predict_train)
recall_cv=recall_score(y_cv,predict_cv)
recall_test=recall_score(y_test,predict_test)

  F1Score_train=f1_score(y_train,predict_train)
  F1Score_cv=f1_score(y_cv,predict_cv)
  F1Score_test=f1_score(y_test,predict_test)

  x = PrettyTable()
  x.field_names = ["Metric", "Train_value" , "CV_value","Test_value"]
```

```
 x.add_row([ "Log-
Loss" ,  np.round(logloss_train,3)  ,  np.round(logloss_cv,3)   ,  np.round(loglos
s_test,3) ])
 x.add_row([ "Precision" , np.round(precision_train,3) ,  np.round(precision_cv
,3)  , np.round(precision_test,3) ])
 x.add_row([ "Recall" ,   np.round(recall_train,3) ,    np.round(recall_cv,3)    ,
np.round(recall_test,3)])
 x.add_row([ "F1-
Score" ,  np.round(F1Score_train,3) ,   np.round(F1Score_cv,3)    , np.round(F1
Score_test,3)])
 x.add_row([ "Accuracy" ,   np.round(accuracy_train,3)  ,  np.round(accuracy_c
v,3)   ,  np.round(accuracy_test,3) ])


 print(x)


 print("Starting Get Important Features Function")
get_imp_features(clf,kernel_select)
```

FUNCTION TO PRINT IMPORTANT FEATURES





CODING:

def get_imp_features(clf,kernel_select):

print(kernel_select) if((kernel_select == 'rbf') or (kernel_select == 'poly')): print(" Kernel is {} so, we cannot get the important Features using Coef_ Funtion".format(kernel_select))

```
  else:

    coefs=sorted(zip(clf.coef_[0],X_train.columns.tolist()))
feat=X_train.columns.tolist()

    top10Negative=coefs[:10]     top10Postive=coefs[::-
1][:10]

    res_neg=pd.DataFrame(top10Negative,columns=['Values','Top10NegativeFea
tures'])
    res_pos=pd.DataFrame(top10Postive,columns=['Values','Top10PostiveFeatur
es'])
    res=pd.concat([res_neg,res_pos],axis=1)

    print("*"*20)
    #print(len(feat))
    #print(len(coefs))

    feat=[i[1] for i in coefs]
coefs1=[i[0] for i in coefs]
plt.figure(figsize=(10,14))
    plt.barh(range(len(feat)), coefs1, align='center')
plt.yticks(range(len(feat)), feat)

    plt.show()

  #return res
```

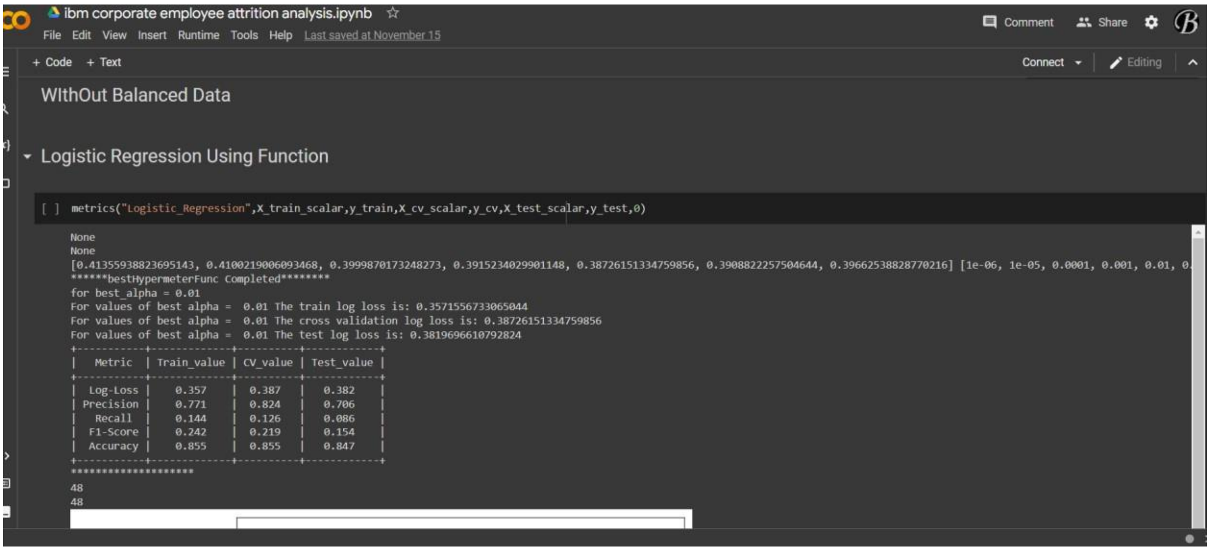
STANDARD SCALAR FUNCTION

CODING:

```
from sklearn.preprocessing import StandardScaler


scaler = StandardScaler()
scaler.fit(X_train)
X_train_scalar=scaler.transform(X_train)
X_cv_scalar=scaler.transform(X_cv)
X_test_scalar=scaler.transform(X_test)
```

WITH OUT BALANCED DATA

LOGISTIC REGRESSION USING FUNCTION



CODING:

None

None

[0.41355938823695143, 0.4100219006093468, 0.3999870173248273, 0.3915234029901148, 0.38726151334759856, 0.3908822257504644, 0.39662538828770216] [1e-06, 1e-05, 0.0001, 0.001, 0.01, 0.1,

1] ******bestHypermeterFunc Completed******** for

best_alpha = 0.01

For values of best alpha = 0.01 The train log loss is: 0.3571556733065044

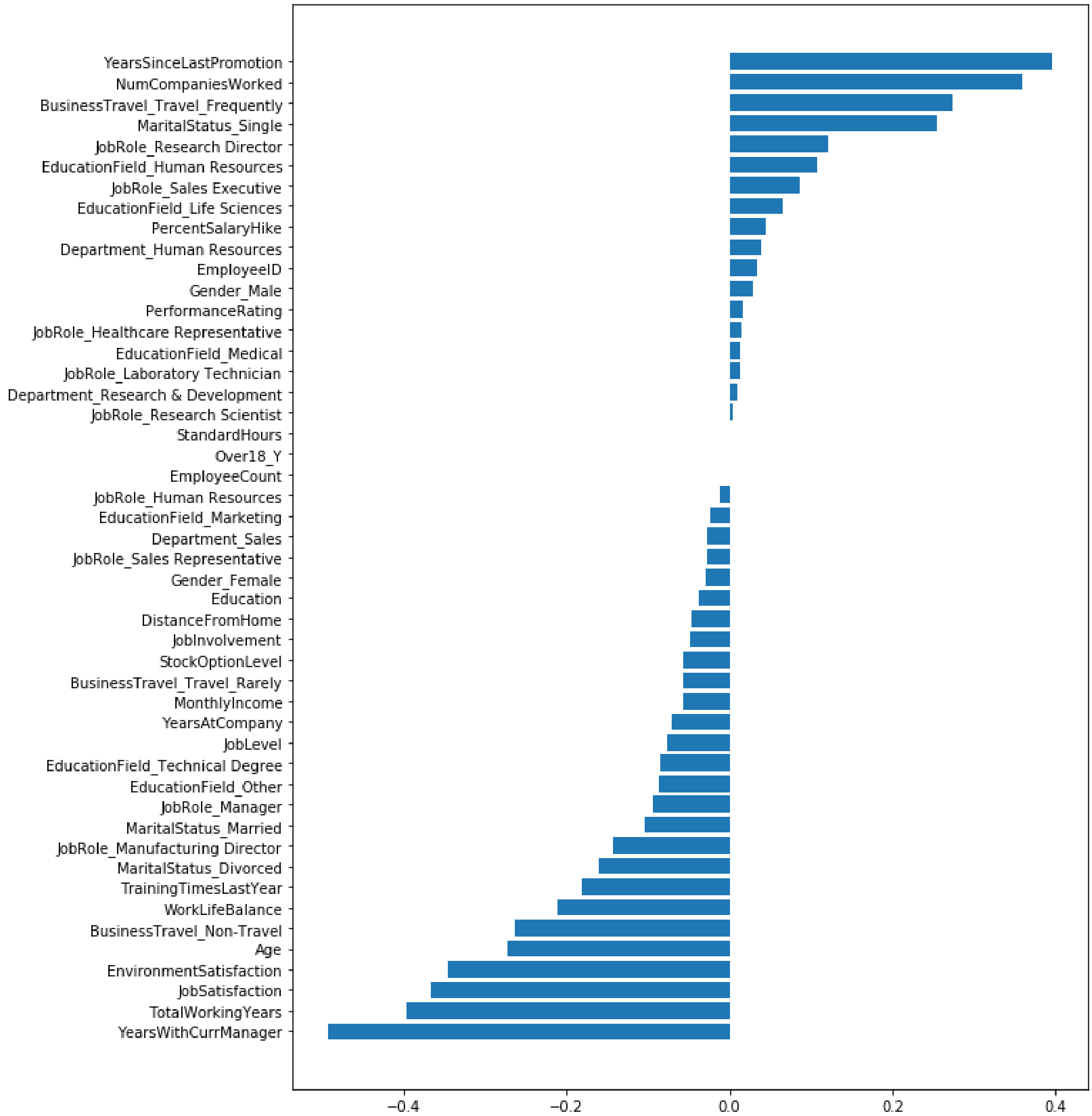
For values of best alpha = 0.01 The cross validation log loss is: 0.38726151334759856

For values of best alpha = 0.01 The test log loss is: 0.3819696610792824

```
+-----------+-------------+----------+------------+
|   Metric  | Train_value | CV_value | Test_value |
+-----------+-------------+----------+------------+
|  Log-Loss |    0.357    |  0.387   |   0.382    |
| Precision |    0.771    |  0.824   |   0.706    |
|   Recall  |    0.144    |  0.126   |   0.086    |
|  F1-Score |    0.242    |  0.219   |   0.154    |
```

```
| Accuracy |    0.855    | 0.855   |   0.847    |
+-----------+-------------+----------+------------+
********************
48
48
```

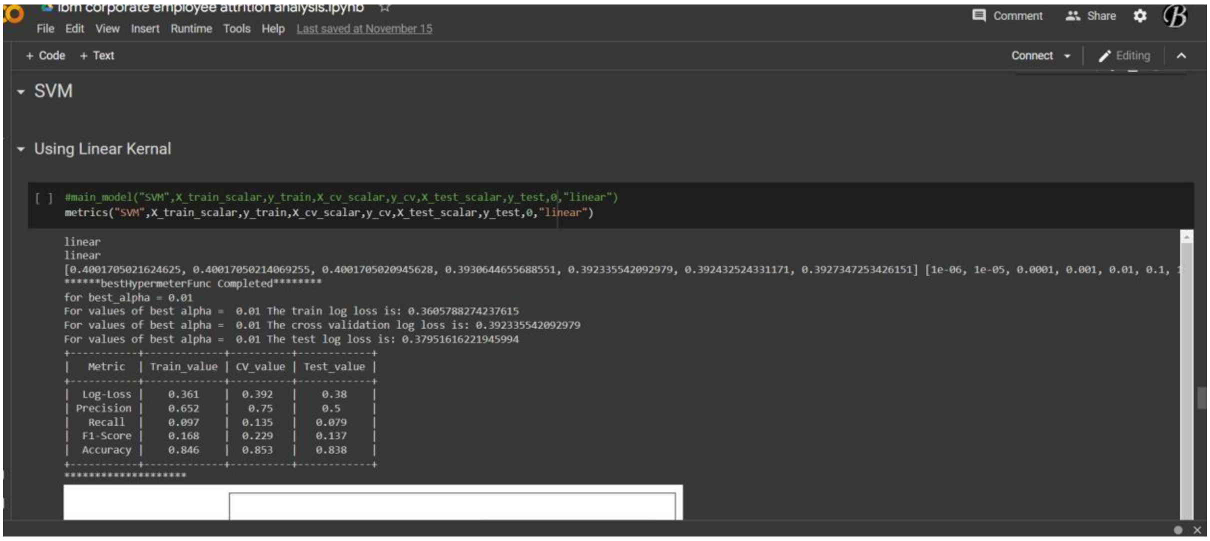
OUTPUT:



SVM

WITH BALANCED DATA:

FOR BALANCING DATA USING SMOTE FUNCTION FROM IMBLEARN PACKAGE



CODING:

linear

linear

[0.4001705021624625, 0.40017050214069255, 0.4001705020945628, 0.3930644655688551, 0.392335542092979, 0.392432524331171, 0.3927347253426151] [1e-06, 1e-05, 0.0001, 0.001, 0.01, 0.1,

1] ******bestHypermeterFunc Completed******** for

best_alpha = 0.01

For values of best alpha = 0.01 The train log loss is: 0.3605788274237615

For values of best alpha = 0.01 The cross validation log loss is: 0.392335542092979

For values of best alpha = 0.01 The test log loss is: 0.37951616221945994

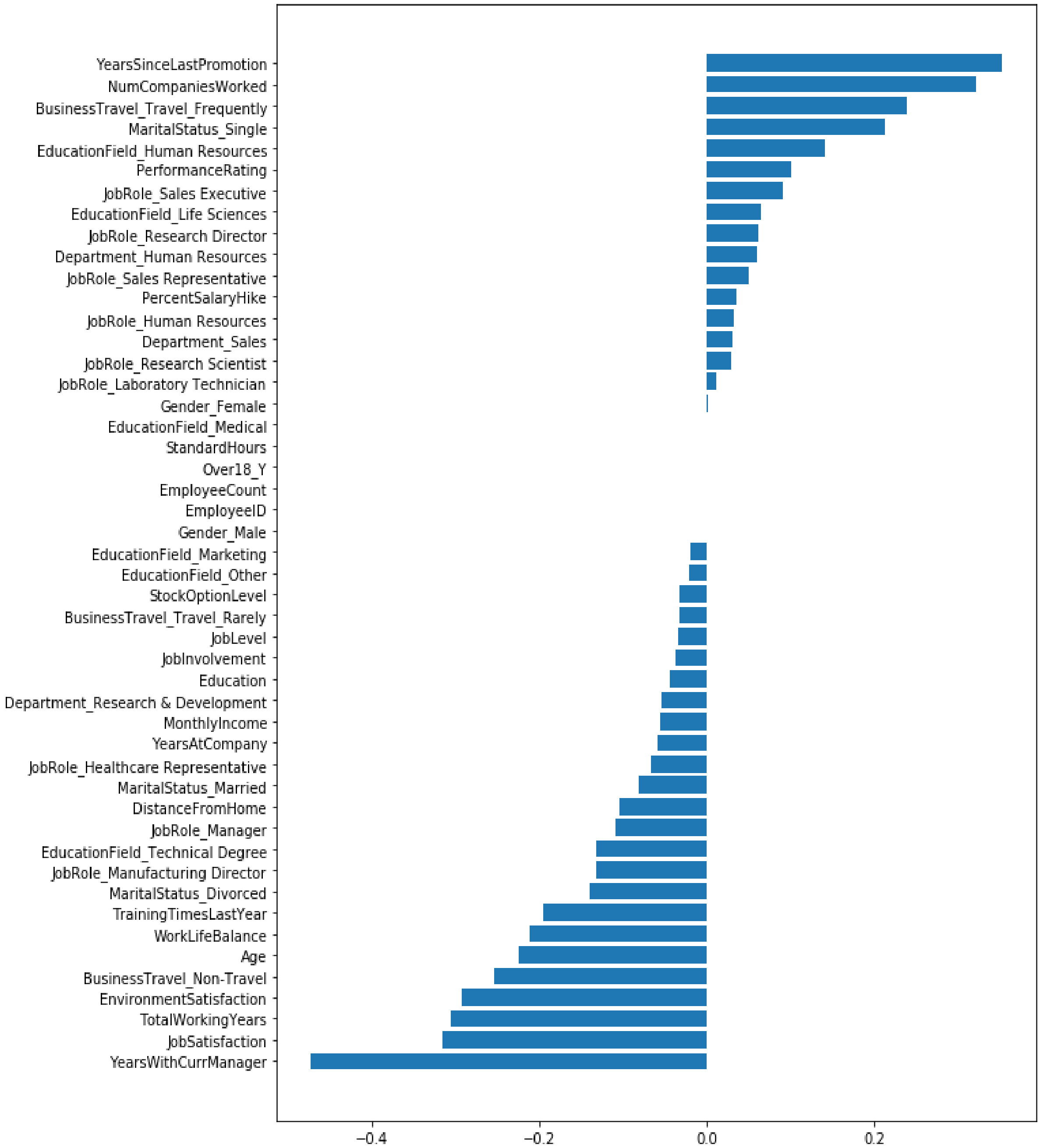
Metric	Train_value	CV_value	Test_value
Log-Loss	0.361	0.392	0.38
Precision	0.652	0.75	0.5
Recall	0.097	0.135	0.079
F1-Score	0.168	0.229	0.137

| Accuracy | 0.846 | 0.853 | 0.838 |

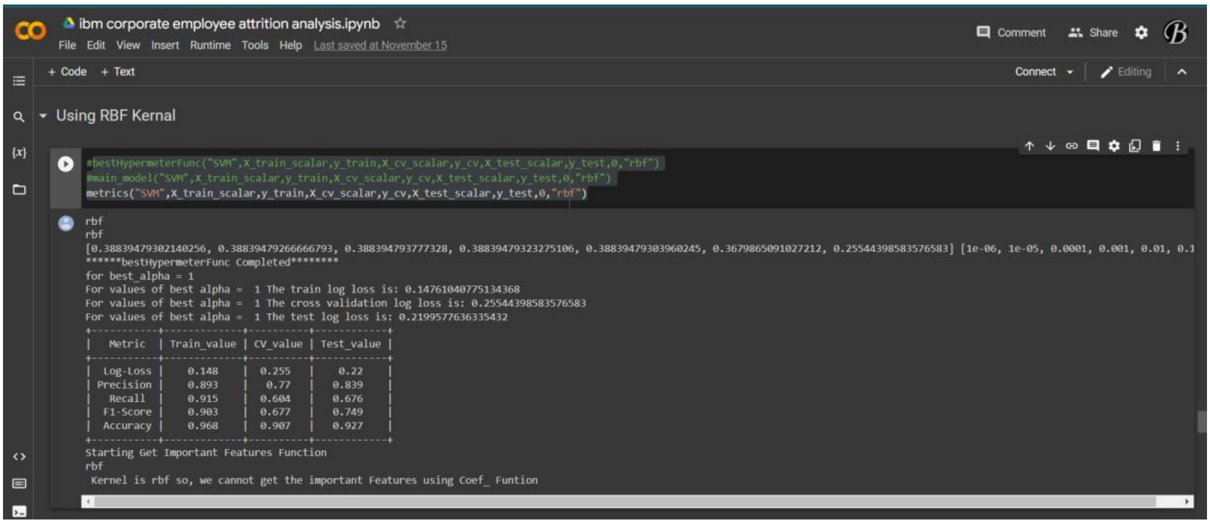
+-----------+-------------+----------+------------+

****

OUTPUT:



USING RBF KERNAL



bestHypermeterFunc("SVM",X_train_scalar,y_train,X_cv_scalar,y_cv,X_test_sca l ar,y_test,0,"rbf")

#main_model("SVM",X_train_scalar,y_train,X_cv_scalar,y_cv,X_test_scalar,y_te st,0,"rbf")

metrics("SVM",X_train_scalar,y_train,X_cv_scalar,y_cv,X_test_scalar,y_test,0,

"r bf") rbf rbf

[0.38839479302140256, 0.38839479266666793, 0.388394793777328, 0.38839479323275106, 0.38839479303960245, 0.3679865091027212, 0.25544398583576583] [1e-06, 1e-05, 0.0001, 0.001, 0.01, 0.1,

1] ******bestHypermeterFunc Completed******** for

best_alpha = 1

For values of best alpha = 1 The train log loss is: 0.14761040775134368

For values of best alpha = 1 The cross validation log loss is: 0.25544398583576583

For values of best alpha = 1 The test log loss is: 0.2199577636335432

```
+-----------+-------------+----------+------------+

|   Metric  | Train_value | CV_value | Test_value |

+-----------+-------------+----------+------------+

|  Log-Loss |    0.148    |  0.255   |    0.22    |

| Precision |    0.893    |   0.77   |   0.839    |

|   Recall  |    0.915    |  0.604   |   0.676    |
```

| F1-Score | 0.903 | 0.677 | 0.749 |

| Accuracy | 0.968 | 0.907 | 0.927 |

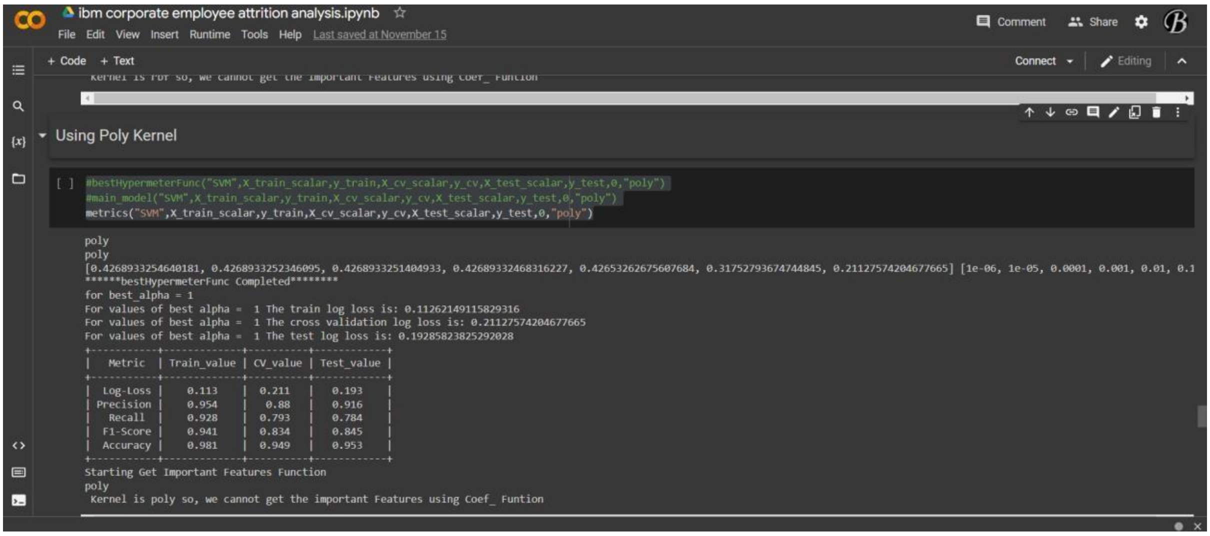
+-----------+-------------+----------+------------+

Starting Get Important Features Function

rbf

Kernel is rbf so, we cannot get the important Features using Coef_ Funtion

USING POLY KERNEL



CODING:

#bestHypermeterFunc("SVM",X_train_scalar,y_train,X_cv_scalar,y_cv,X_test_scalar,y_test,0,"poly")

#main_model("SVM",X_train_scalar,y_train,X_cv_scalar,y_cv,X_test_scalar,y_te st,0,"poly") poly poly

[0.4268933254640181, 0.4268933252346095, 0.4268933251404933, 0.42689332468316227, 0.42653262675607684, 0.31752793674744845, 0.21127574204677665] [1e-06, 1e-05, 0.0001, 0.001, 0.01, 0.1,

1] ******bestHypermeterFunc Completed******** for

best_alpha = 1

For values of best alpha = 1 The train log loss is: 0.11262149115829316

For values of best alpha = 1 The cross validation log loss is: 0.21127574204677665

For values of best alpha = 1 The test log loss is: 0.19285823825292028

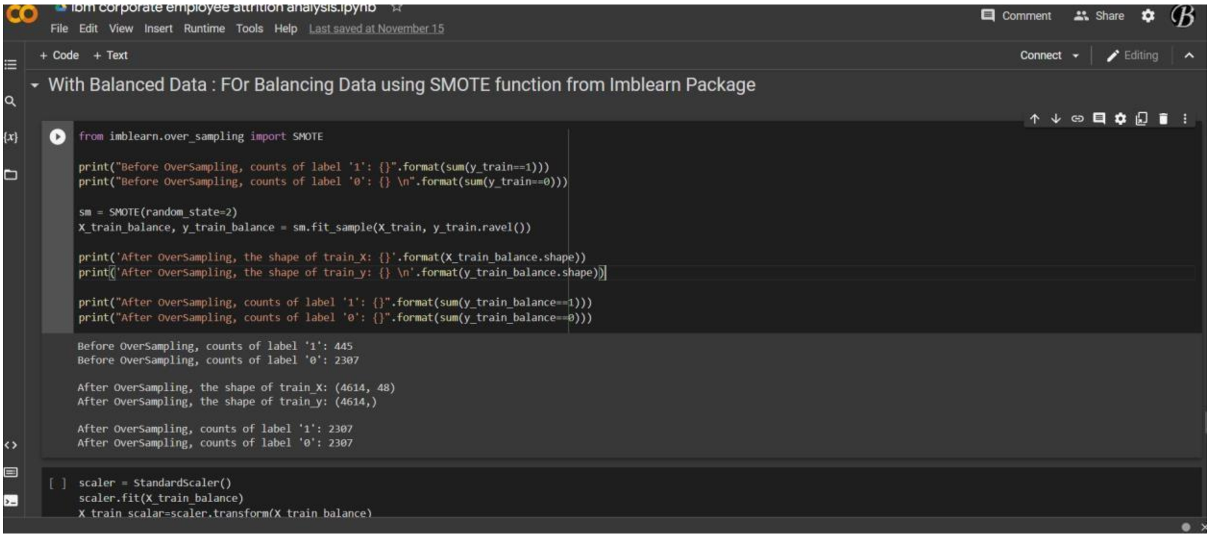
Metric	Train_value	CV_value	Test_value
Log-Loss	0.113	0.211	0.193
Precision	0.954	0.88	0.916
Recall	0.928	0.793	0.784
F1-Score	0.941	0.834	0.845
Accuracy	0.981	0.949	0.953

Starting Get Important Features Function

poly

Kernel is poly so, we cannot get the important Features using Coef_
Funtionmetrics("SVM",X_train_scalar,y_train,X_cv_scalar,y_cv,X_test_scalar,y_test,0,"poly

WITH BALANCED DATA: FOR BALANCING DATA USING SMOTE FUNCTION FROM IMBLEARN PACKAGE



CODING:

```
from imblearn.over_sampling import SMOTE


print("Before OverSampling, counts of label '1': {}".format(sum(y_train==1)))

print("Before OverSampling, counts of label '0': {} \n".format(sum(y_train==0)))


sm = SMOTE(random_state=2)

X_train_balance, y_train_balance = sm.fit_sample(X_train, y_train.ravel())


print('After OverSampling, the shape of train_X:
{}'.format(X_train_balance.shap e))

print('After OverSampling, the shape of train_y: {}
\n'.format(y_train_balance.sh ape))


print("After OverSampling, counts of label '1':
{}".format(sum(y_train_balance==
1)))

print("After OverSampling, counts of label '0':
{}".format(sum(y_train_balance== 0)))
```

OUTPUT:

Before OverSampling, counts of label '1': 445

Before OverSampling, counts of label '0': 2307

After OverSampling, the shape of train_X: (4614, 48)

After OverSampling, the shape of train_y: (4614,)

After OverSampling, counts of label '1': 2307

After OverSampling, counts of label '0': 2307

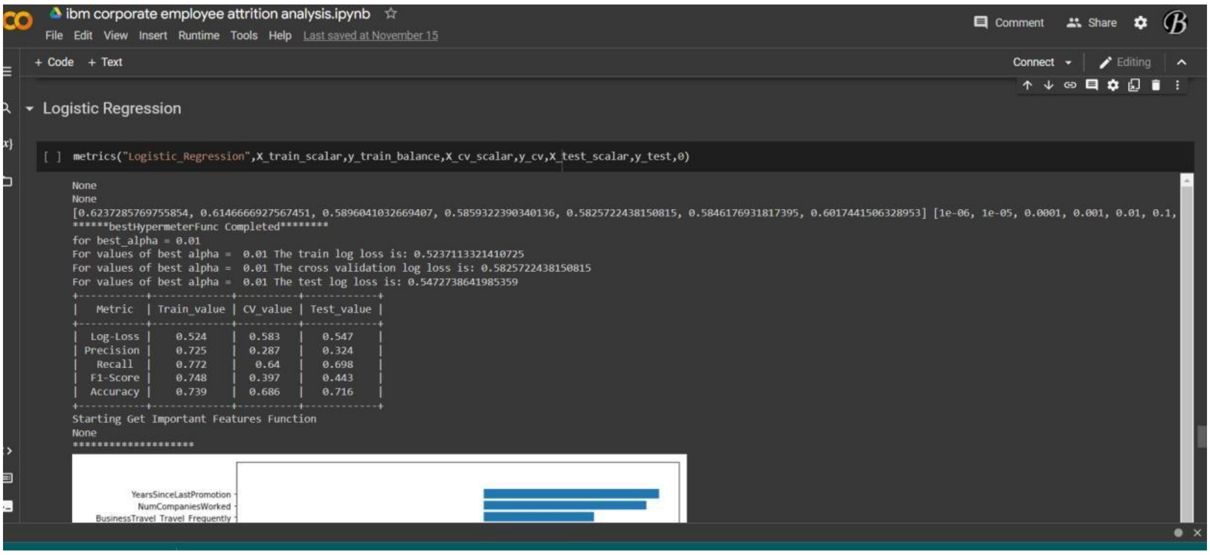
CODING:

```
scaler = StandardScaler()
scaler.fit(X_train_balance)
X_train_scalar=scaler.transform(X_train_balance)
X_cv_scalar=scaler.transform(X_cv)
X_test_scalar=scaler.transform(X_test)

X_train_scalar.shape
```

OUTPUT:

(4614, 48)

LOGISTIC REGRESSION



CODING:

None

None

[0.6237285769755854, 0.6146666927567451, 0.5896041032669407, 0.5859322390340136, 0.5825722438150815, 0.5846176931817395, 0.6017441506328953] [1e-06, 1e-05, 0.0001, 0.001, 0.01, 0.1,

1] ******bestHypermeterFunc Completed******** for

best_alpha = 0.01

For values of best alpha = 0.01 The train log loss is: 0.5237113321410725

For values of best alpha = 0.01 The cross validation log loss is: 0.5825722438150815

For values of best alpha = 0.01 The test log loss is: 0.5472738641985359

```
+-----------+-------------+----------+------------+
|   Metric  | Train_value | CV_value | Test_value |
+-----------+-------------+----------+------------+
|  Log-Loss |    0.524    |  0.583   |   0.547    |
| Precision |    0.725    |  0.287   |   0.324    |
|   Recall  |    0.772    |   0.64   |   0.698    |
|  F1-Score |    0.748    |  0.397   |   0.443    |
```

| Accuracy | 0.739 | 0.686 | 0.716 |

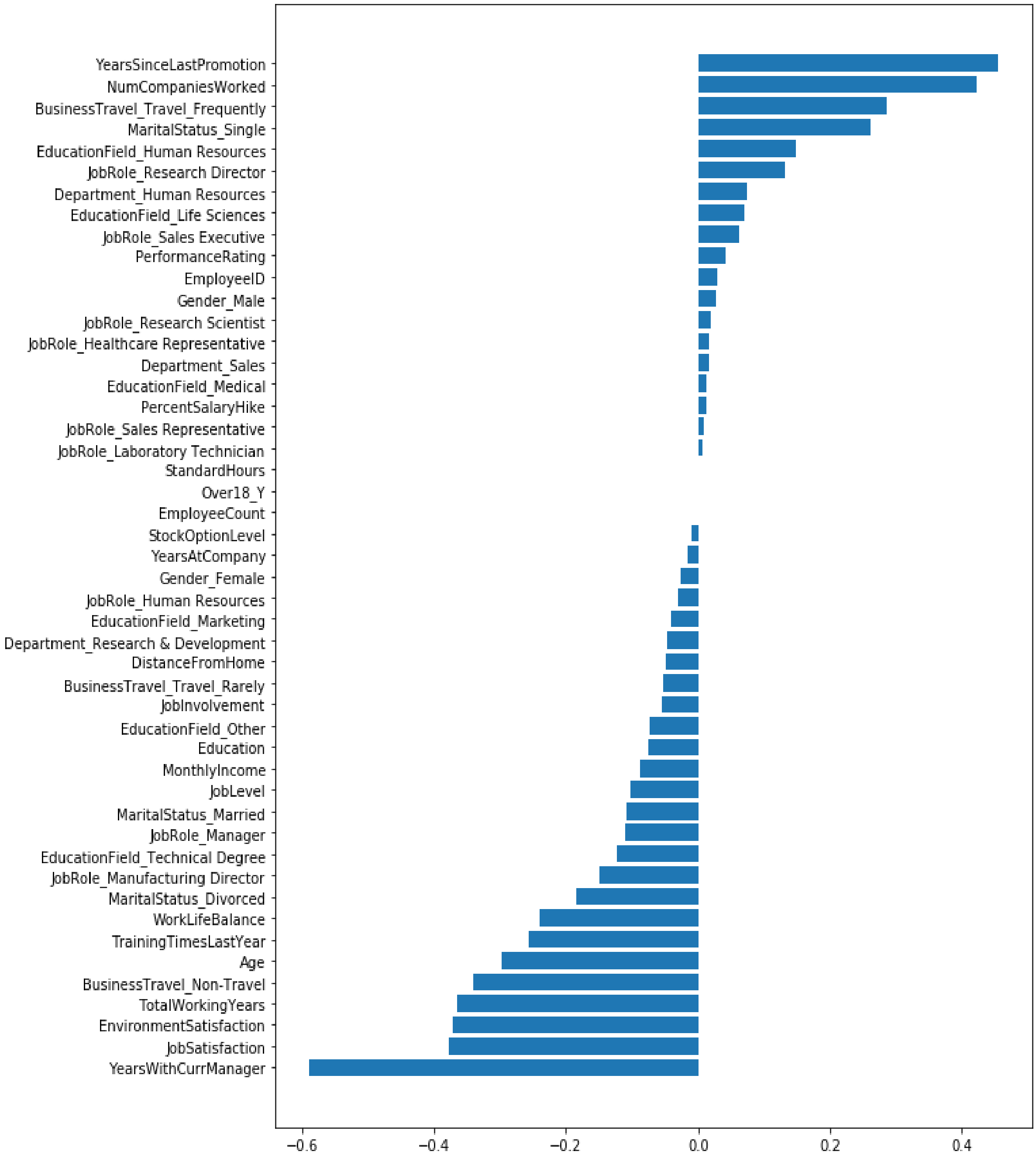
+-----------+-------------+----------+------------+

Starting Get Important Features Function

None

****

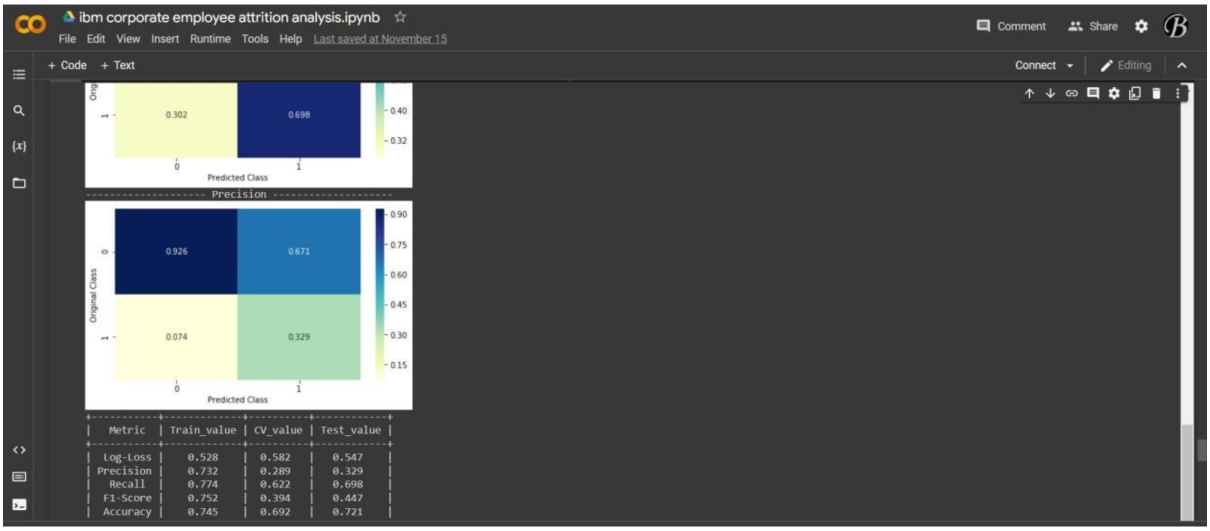
OUTPUT:

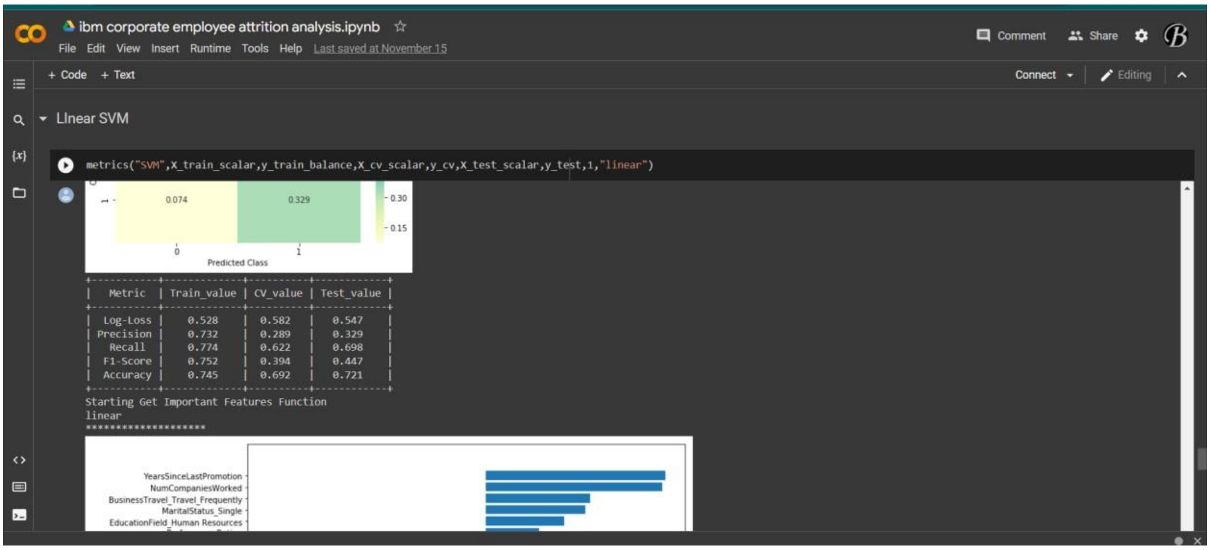


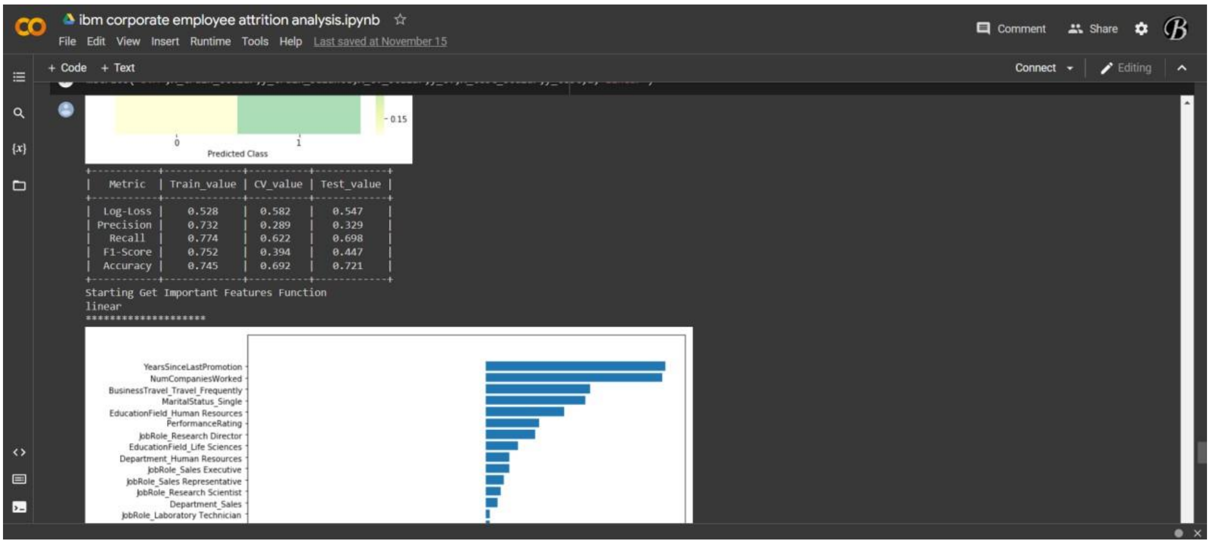
SVM

LINEAR SVM


ibm corporate employee attrition analysis.ipynb
File Edit View Insert Runtime Tools Help Last saved at November 15
Comment Share
+ Code + Text
Connect Editing
SVM
Linear SVM
metrics("SVM",X_train_scalar,y_train_balance,X_cv_scalar,y_cv,X_test_scalar,y_test,1,"linear")
Log Loss : 0.6142136356700172
for alpha = 0.001
Log Loss : 0.5871537845084024
for alpha = 0.01
Log Loss : 0.5817866567511131
for alpha = 0.1
Log Loss : 0.583009140383031
for alpha = 1
Log Loss : 0.5832164963645
Cross Validation Error for each alpha
Error measure
0.615
0.610
0.605
0.600
0.595
0.590
(-6.907755278982137, '0.58')


ibm corporate employee attrition analysis.ipynb
File Edit View Insert Runtime Tools Help Last saved at November 15
Comment Share
+ Code + Text
Connect Editing
0.302
0.698
0.40
0.32
Predicted Class
Precision
0.926
0.671
0.074
0.329
0.90
0.75
0.60
0.45
0.30
0.15
Original Class
Predicted Class
Metric	Train_value	CV_value	Test_value
Log-Loss	0.528	0.582	0.547
Precision	0.732	0.289	0.329
Recall	0.774	0.622	0.698
F1-Score	0.752	0.394	0.447
Accuracy	0.745	0.692	0.721


ibm corporate employee attrition analysis.ipynb
File Edit View Insert Runtime Tools Help Last saved at November 15
Comment Share
+ Code + Text
Connect Editing
Linear SVM
metrics("SVM",X_train_scalar,y_train_balance,X_cv_scalar,y_cv,X_test_scalar,y_test,1,"linear")
0.074
0.329
0.30
0.15
Predicted Class
Metric	Train_value	CV_value	Test_value
Log-Loss	0.528	0.582	0.547
Precision	0.732	0.289	0.329
Recall	0.774	0.622	0.698
F1-Score	0.752	0.394	0.447
Accuracy	0.745	0.692	0.721
Starting Get Important Features Function
linear
YearsSinceLastPromotion
NumCompaniesWorked
BusinessTravel_Travel_Frequently
MaritalStatus_Single
EducationField_Human Resources



CODING:

```
metrics("SVM",X_train_scalar,y_train_balance,X_cv_scalar,y_cv,X_test_scalar,y_test,1,"linear")
```

OUTPUT:

linear for alpha

= 1e-06

Log Loss : 0.6143143363763562

for alpha = 1e-05

Log Loss : 0.6143143337374273

for alpha = 0.0001

Log Loss : 0.6142136356700172

for alpha = 0.001

Log Loss : 0.5871537845084024

for alpha = 0.01

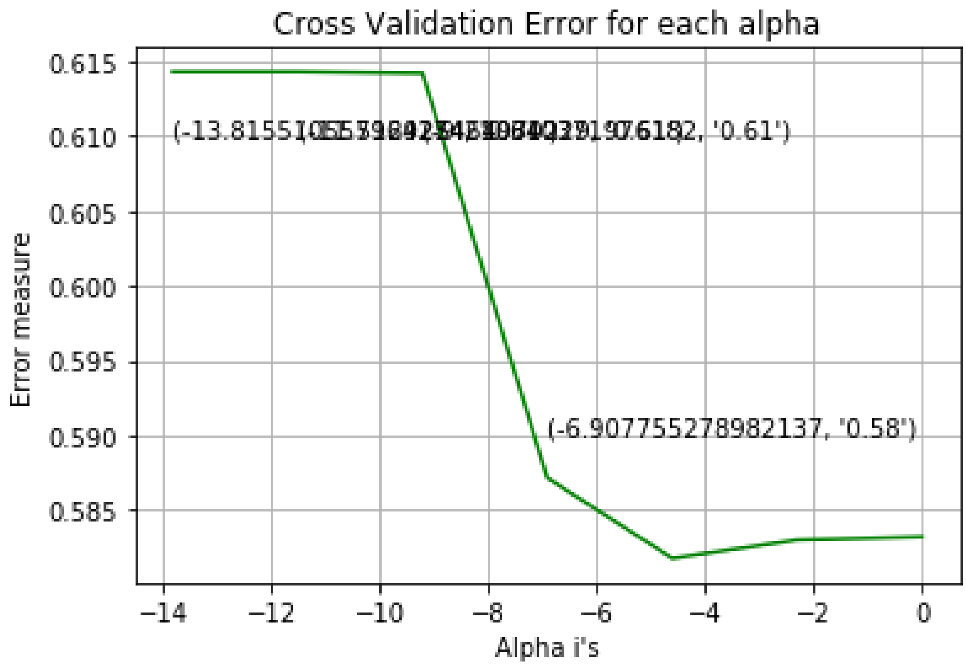
Log Loss : 0.5817866567511131

for alpha = 0.1

Log Loss : 0.583009140383031

for alpha = 1

Log Loss : 0.5832164963645



linear

[0.6143143363763562, 0.6143143337374273, 0.6142136356700172, 0.5871537845084024, 0.5817866567511131, 0.583009140383031, 0.5832164963645] [1e-06, 1e-05, 0.0001, 0.001, 0.01, 0.1,

1] ******bestHypermeterFunc Completed******** for

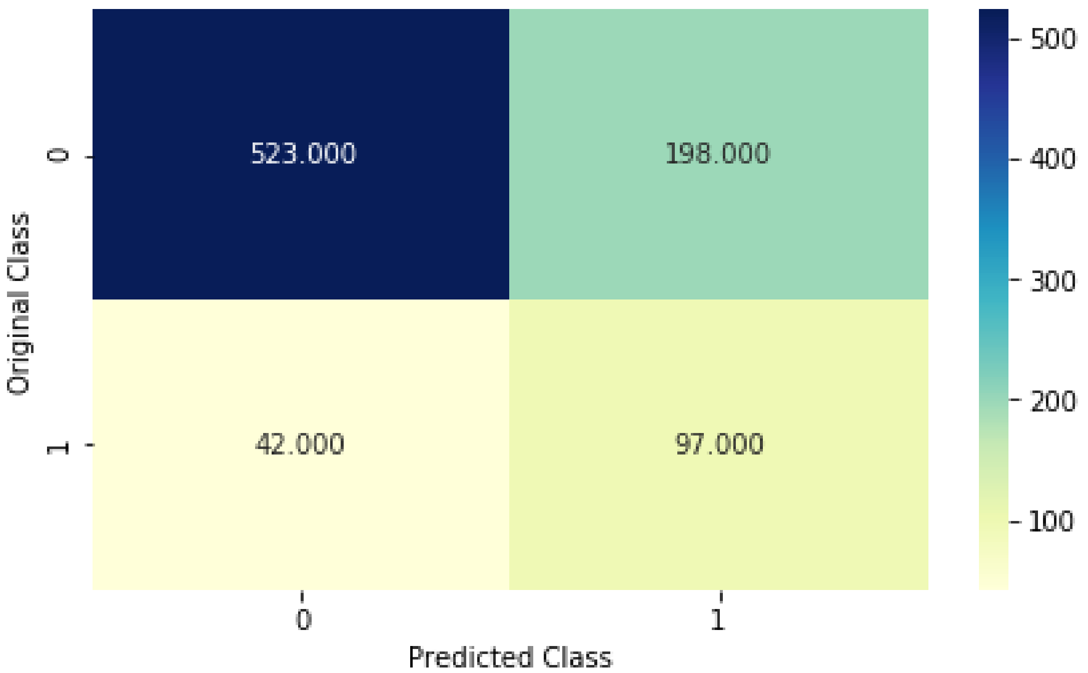
best_alpha = 0.01

For values of best alpha = 0.01 The train log loss is: 0.5278361295624395

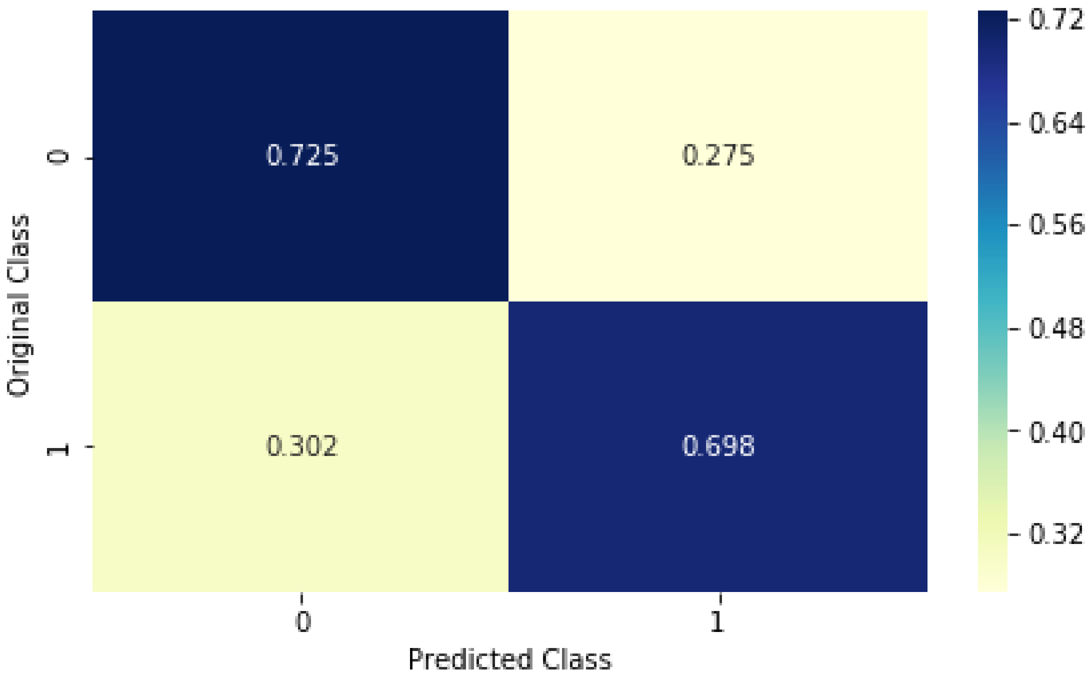
For values of best alpha = 0.01 The cross validation log loss is: 0.5817866567511131

For values of best alpha = 0.01 The test log loss is: 0.5474462705866496

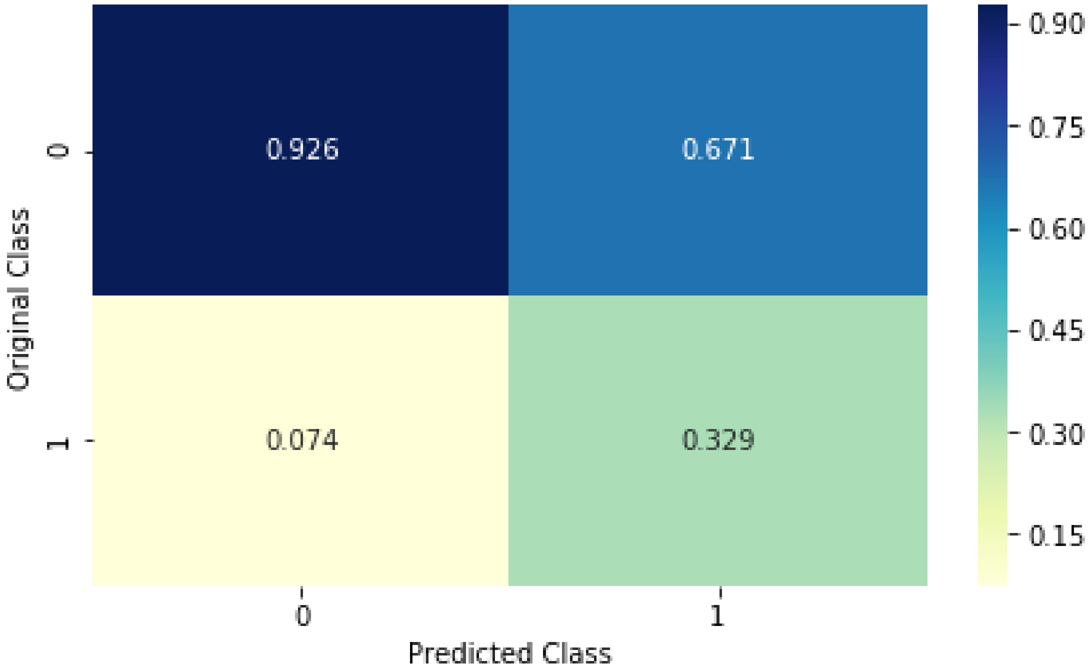
------------------- Confusion Matrix --------------------


523.000
198.000
42.000
97.000
Original Class
0
1
Predicted Class
500
400
300
200
100

-------------------- Recall --------------------


0.725
0.275
0.302
0.698
Original Class
0
1
Predicted Class
0.72
0.64
0.56
0.48
0.40
0.32

------------------- Precision -------------------

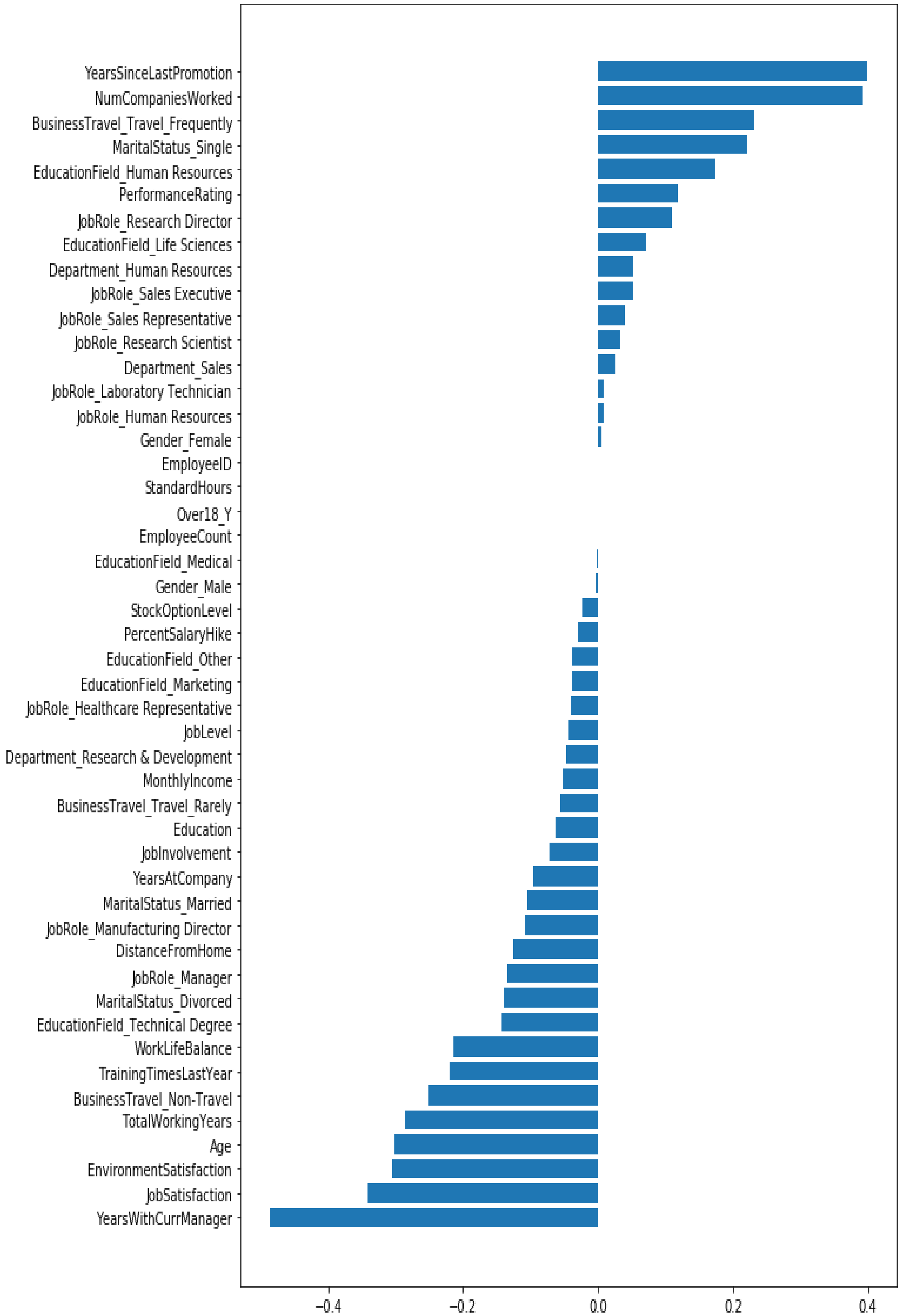


Metric	Train_value	CV_value	Test_value
Log-Loss	0.528	0.582	0.547
Precision	0.732	0.289	0.329
Recall	0.774	0.622	0.698
F1-Score	0.752	0.394	0.447
Accuracy	0.745	0.692	0.721

Starting Get Important Features Function

linear

****


YearsSinceLastPromotion
NumCompaniesWorked
BusinessTravel_Travel_Frequently
MaritalStatus_Single
EducationField_Human Resources
PerformanceRating
JobRole_Research Director
EducationField_Life Sciences
Department_Human Resources
JobRole_Sales Executive
JobRole_Sales Representative
JobRole_Research Scientist
Department_Sales
JobRole_Laboratory Technician
JobRole_Human Resources
Gender_Female
EmployeeID
StandardHours
Over18_Y
EmployeeCount
EducationField_Medical
Gender_Male
StockOptionLevel
PercentSalaryHike
EducationField_Other
EducationField_Marketing
JobRole_Healthcare Representative
JobLevel
Department_Research & Development
MonthlyIncome
BusinessTravel_Travel_Rarely
Education
JobInvolvement
YearsAtCompany
MaritalStatus_Married
JobRole_Manufacturing Director
DistanceFromHome
JobRole_Manager
MaritalStatus_Divorced
EducationField_Technical Degree
WorkLifeBalance
TrainingTimesLastYear
BusinessTravel_Non-Travel
TotalWorkingYears
Age
EnvironmentSatisfaction
JobSatisfaction
YearsWithCurrManager
−0.4
−0.2
0.0
0.2
0.4

RBF KERNAL

CODING:

metrics("SVM",X_train_scalar,y_train_balance,X_cv_scalar,y_cv,X_test_scalar,y_test,1,"rbf")

OUTPUT:

rbf

for alpha = 1e-06

Log Loss : 0.5737661950848786

for alpha = 1e-05

Log Loss : 0.5737661982669221

for alpha = 0.0001

Log Loss : 0.5737661920566736

for alpha = 0.001

Log Loss : 0.5737661955475888

for alpha = 0.01

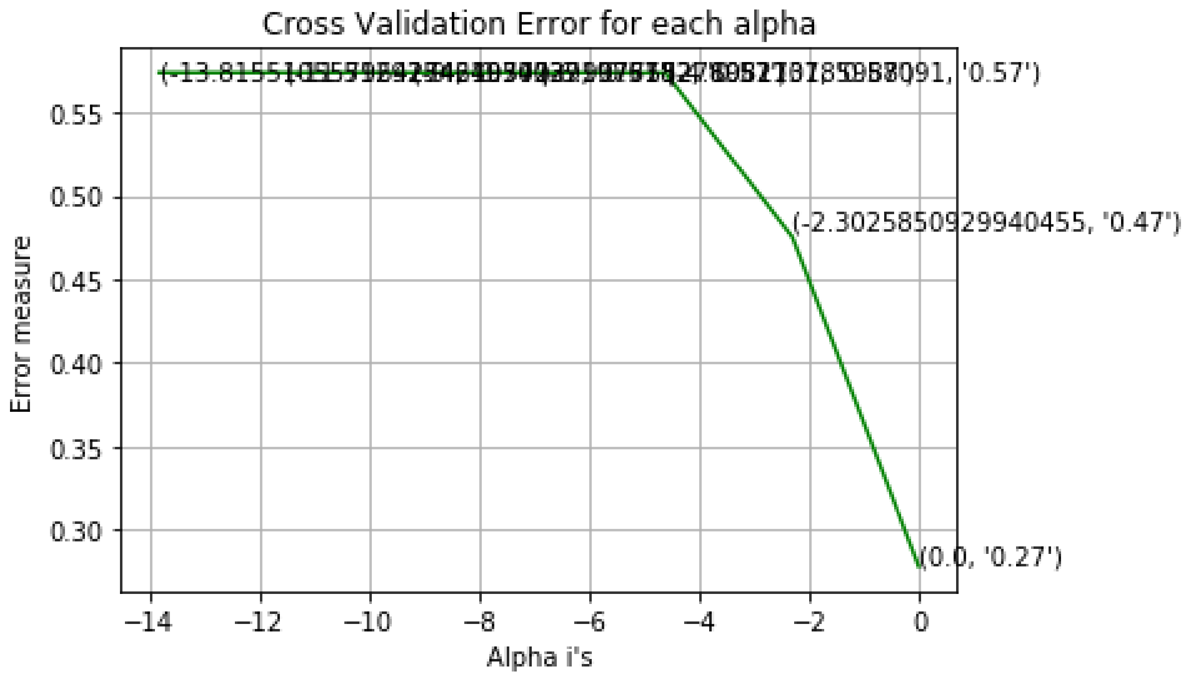
Log Loss : 0.5735611718000113

for alpha = 0.1

Log Loss : 0.47514516397540624

for alpha = 1

Log Loss : 0.27810729173557835



rbf

[0.5737661950848786, 0.5737661982669221, 0.5737661920566736, 0.5737661955475888, 0.5735611718000113, 0.47514516397540624, 0.27810729173557835] [1e-06, 1e-05, 0.0001, 0.001, 0.01, 0.1,

1] ******bestHypermeterFunc Completed******** for

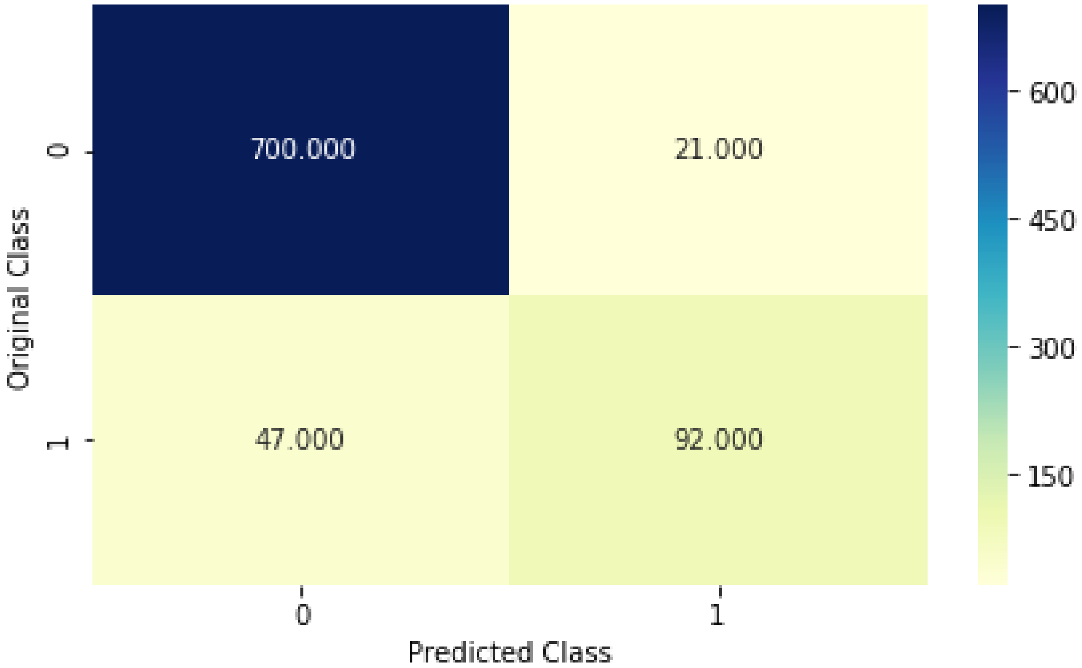
best_alpha = 1

For values of best alpha = 1 The train log loss is: 0.06707151930965102

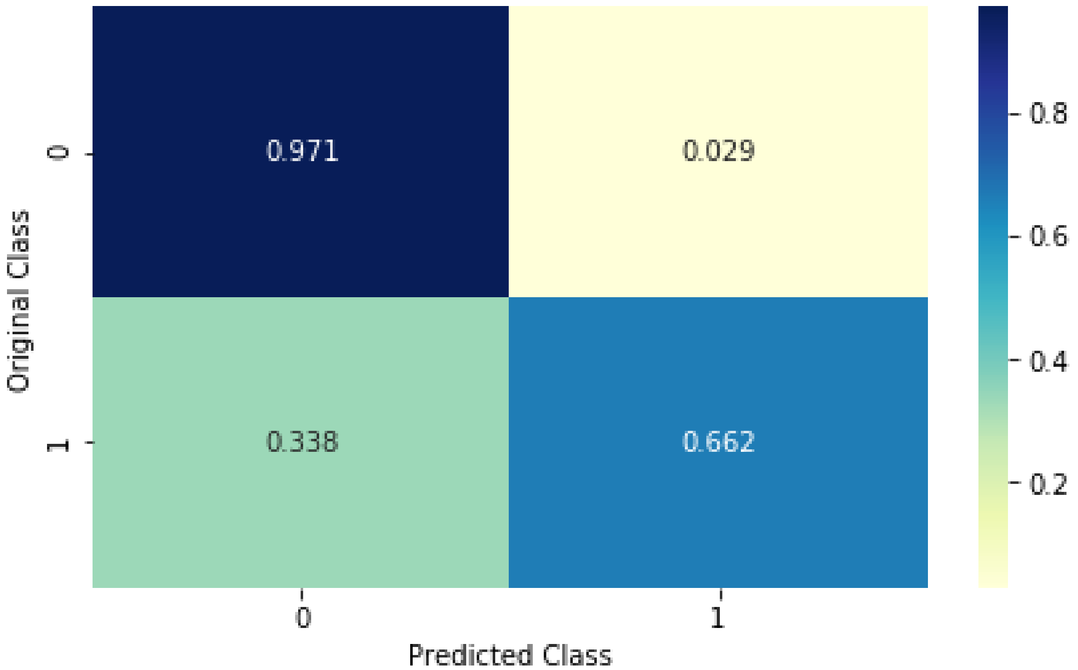
For values of best alpha = 1 The cross validation log loss is: 0.27810729173557835

For values of best alpha = 1 The test log loss is: 0.21711277243059424

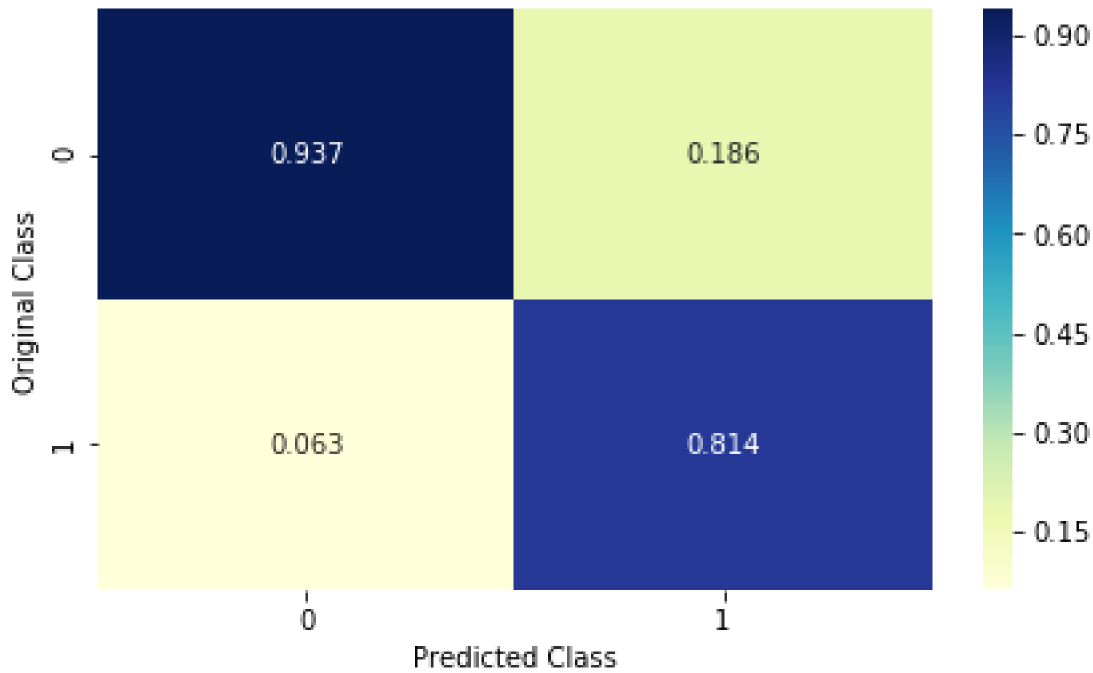
-------------------- Confusion Matrix --------------------


700.000
21.000
47.000
92.000
Original Class
Predicted Class
0
1
0
1
600
450
300
150

-------------------- Recall --------------------


0.971
0.029
0.338
0.662
Original Class
Predicted Class
0
1
0
1
0.8
0.6
0.4
0.2

-------------------- Precision --------------------



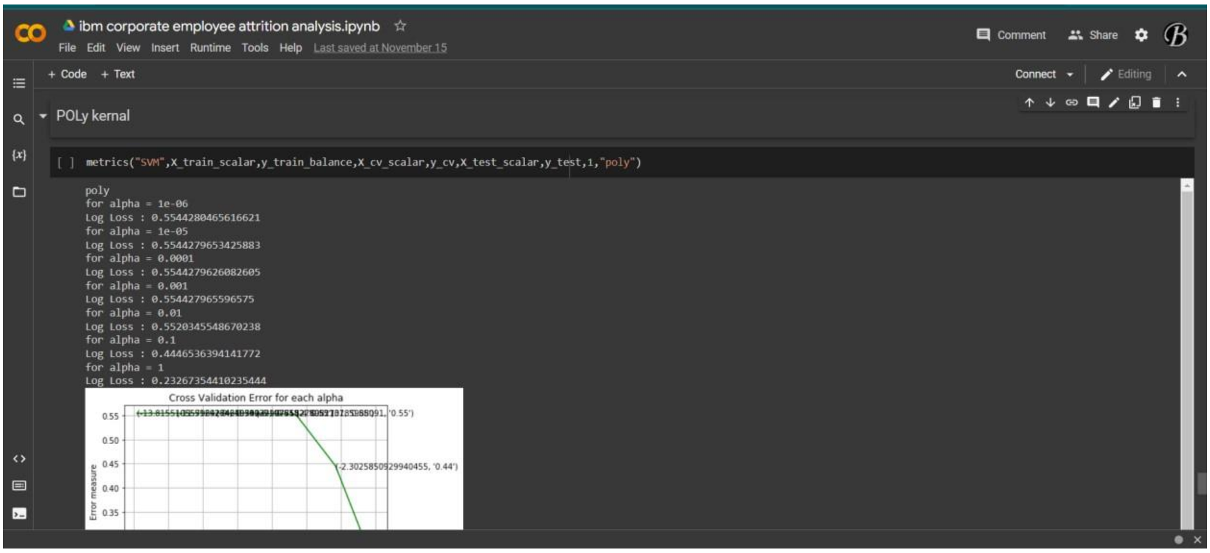
Metric	Train_value	CV_value	Test_value
Log-Loss	0.067	0.278	0.217
Precision	0.985	0.722	0.814
Recall	0.975	0.631	0.662
F1-Score	0.98	0.673	0.73
Accuracy	0.98	0.901	0.921

Starting Get Important Features Function

rbf

Kernel is rbf so, we cannot get the important Features using Coef_ Funtion

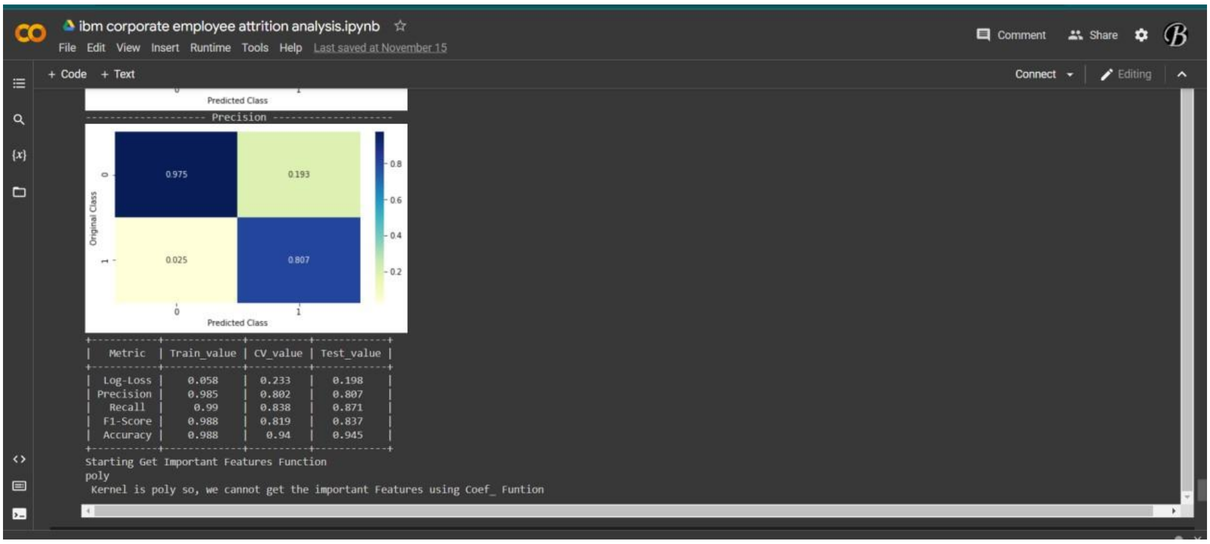
POLY KERNAL



ibm corporate employee attrition analysis.ipynb
File Edit View Insert Runtime Tools Help Last saved at November 15
Comment Share
+ Code + Text
Connect Editing
POLy kernal
metrics("SVM",X_train_scalar,y_train_balance,X_cv_scalar,y_cv,X_test_scalar,y_test,1,"poly")
poly
for alpha = 1e-06
Log Loss : 0.5544280465616621
for alpha = 1e-05
Log Loss : 0.5544279653425883
for alpha = 0.0001
Log Loss : 0.5544279626082605
for alpha = 0.001
Log Loss : 0.554427965596575
for alpha = 0.01
Log Loss : 0.5520345548670238
for alpha = 0.1
Log Loss : 0.4446536394141772
for alpha = 1
Log Loss : 0.23267354410235444
Cross Validation Error for each alpha
(-2.302585092994045, '0.44')
Error measure

ibm corporate employee attrition analysis.ipynb
File Edit View Insert Runtime Tools Help Last saved at November 15
Comment Share
+ Code + Text
Connect Editing
POLy kernal
metrics("SVM",X_train_scalar,y_train_balance,X_cv_scalar,y_cv,X_test_scalar,y_test,1,"poly")
[0.5544280465616621, 0.5544279653425883, 0.5544279626082605, 0.554427965596575, 0.5520345548670238, 0.4446536394141772, 0.23267354410235444] [1e-06, 1e-05, 0.0001, 0.001, 0.01, 0.1,
******bestHypermeterFunc Completed********
for best_alpha = 1
For values of best alpha = 1 The train log loss is: 0.05843126754768401
For values of best alpha = 1 The cross validation log loss is: 0.23267354410235444
For values of best alpha = 1 The test log loss is: 0.19790587725653494
-------------------- Confusion Matrix --------------------
692.000
29.000
18.000
121.000
Original Class
Predicted Class
-------------------- Recall --------------------

ibm corporate employee attrition analysis.ipynb
File Edit View Insert Runtime Tools Help Last saved at November 15
Comment Share
+ Code + Text
Connect Editing
Predicted Class
-------------------- Recall --------------------
0.960
0.040
0.129
0.871
Original Class
Predicted Class
-------------------- Precision --------------------
0.975
0.193
0.025
0.807
Original Class



CODING:

metrics("SVM",X_train_scalar,y_train_balance,X_cv_scalar,y_cv,X_test_scalar,y_test,1,"poly")

OUTPUT:

poly for alpha =

1e-06

Log Loss : 0.5544280465616621

for alpha = 1e-05

Log Loss : 0.5544279653425883

for alpha = 0.0001

Log Loss : 0.5544279626082605

for alpha = 0.001

Log Loss : 0.554427965596575

for alpha = 0.01

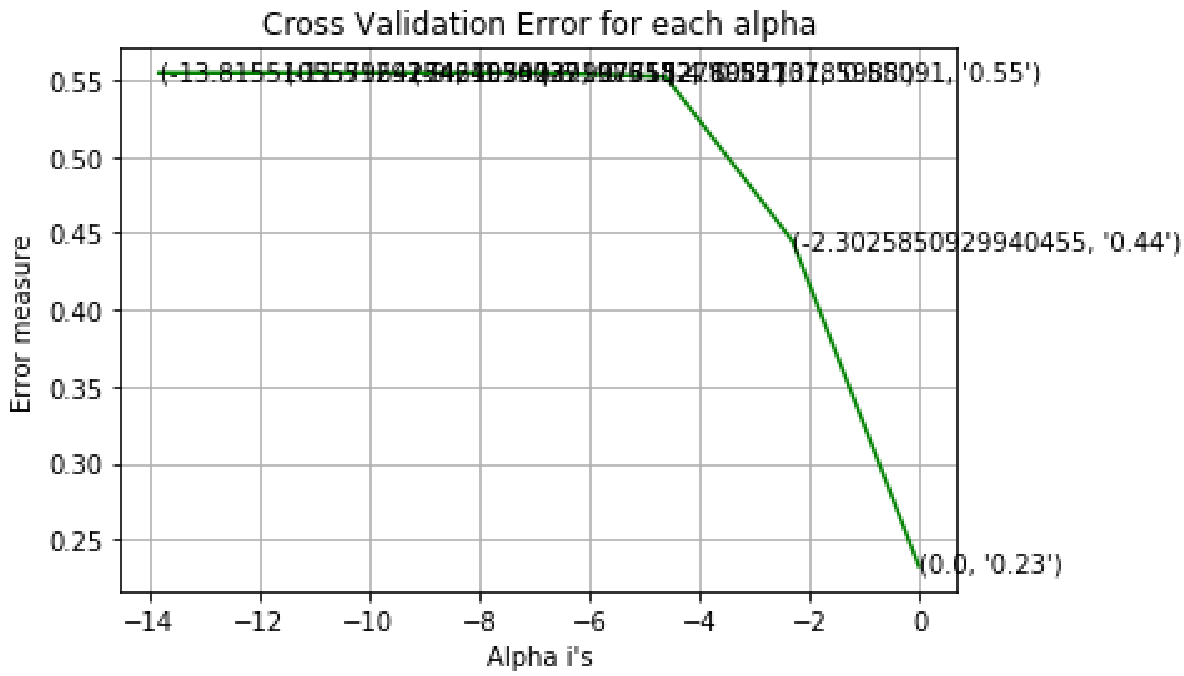
Log Loss : 0.5520345548670238

for alpha = 0.1

Log Loss : 0.4446536394141772

for alpha = 1

Log Loss : 0.23267354410235444



poly

[0.5544280465616621, 0.5544279653425883, 0.5544279626082605, 0.554427965596575, 0.5520345548670238, 0.4446536394141772, 0.23267354410235444] [1e-06, 1e-05, 0.0001, 0.001, 0.01, 0.1,

1] ******bestHypermeterFunc Completed******** for

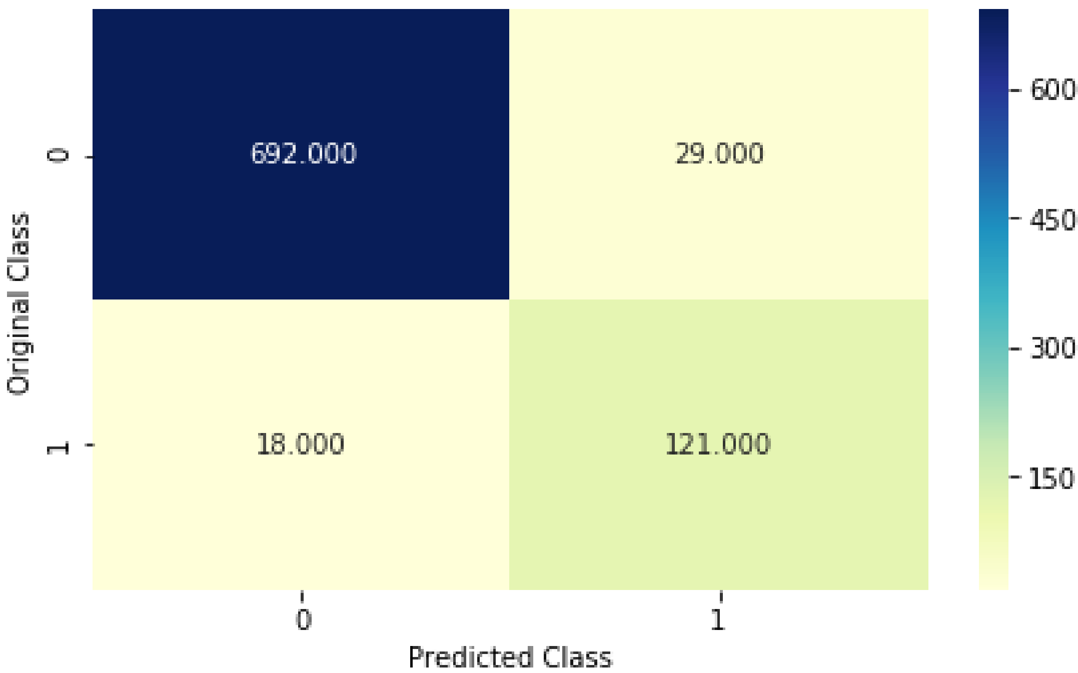
best_alpha = 1

For values of best alpha = 1 The train log loss is: 0.05843126754768401

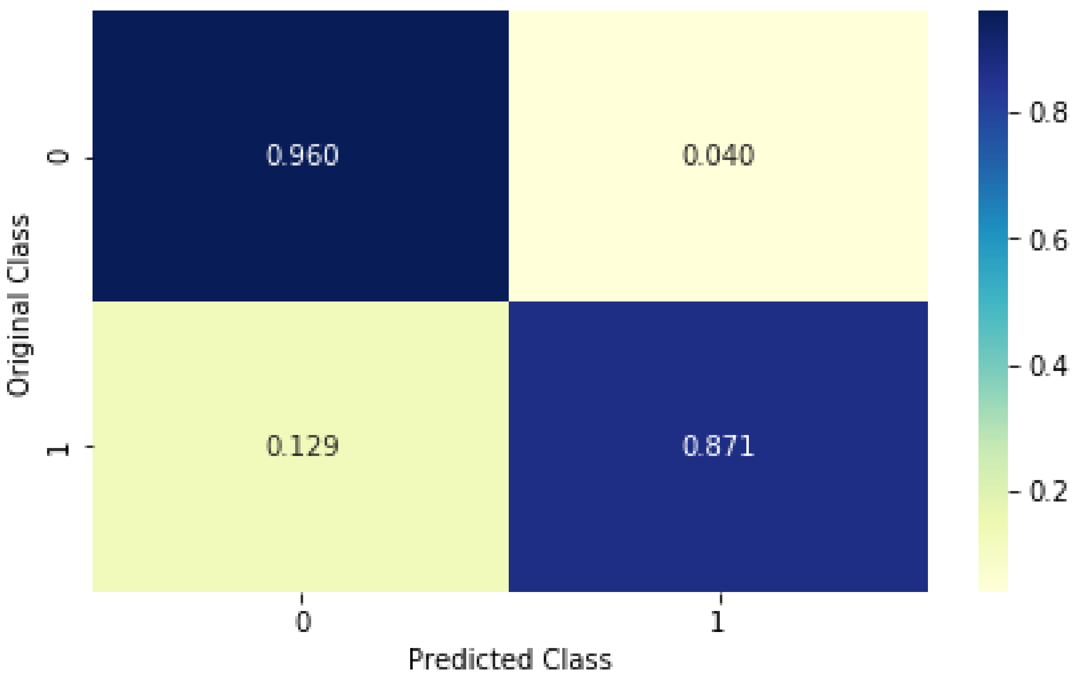
For values of best alpha = 1 The cross validation log loss is: 0.23267354410235444

For values of best alpha = 1 The test log loss is: 0.19790587725653494

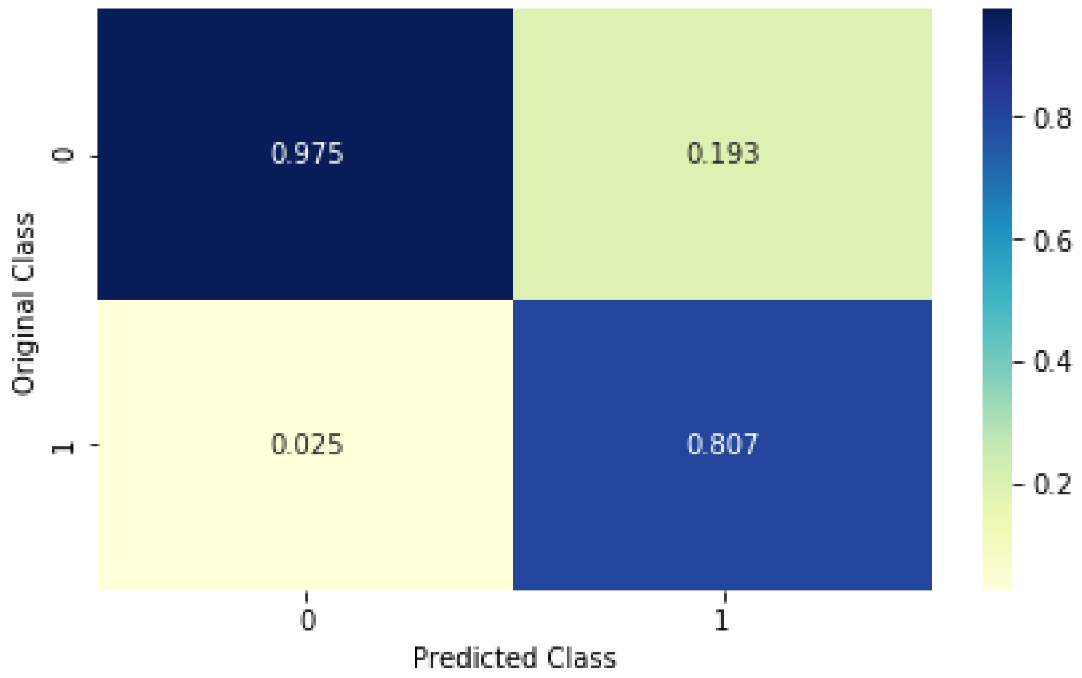
-------------------- Confusion Matrix --------------------


692.000
29.000
18.000
121.000
600
450
300
150
Original Class
0
1
0
1
Predicted Class

-------------------- Recall --------------------


0.960
0.040
0.129
0.871
0.8
0.6
0.4
0.2
Original Class
0
1
0
1
Predicted Class

-------------------- Precision --------------------

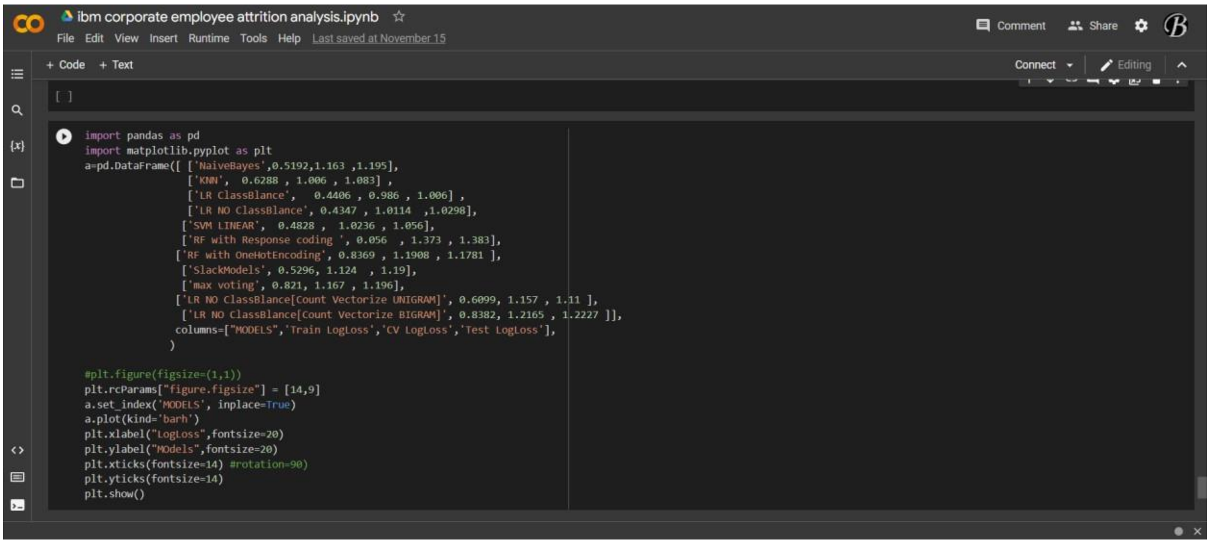


Metric	Train_value	CV_value	Test_value
Log-Loss	0.058	0.233	0.198
Precision	0.985	0.802	0.807
Recall	0.99	0.838	0.871
F1-Score	0.988	0.819	0.837
Accuracy	0.988	0.94	0.945

Starting Get Important Features Function

poly

Kernel is poly so, we cannot get the important Features using Coef_ Funtion

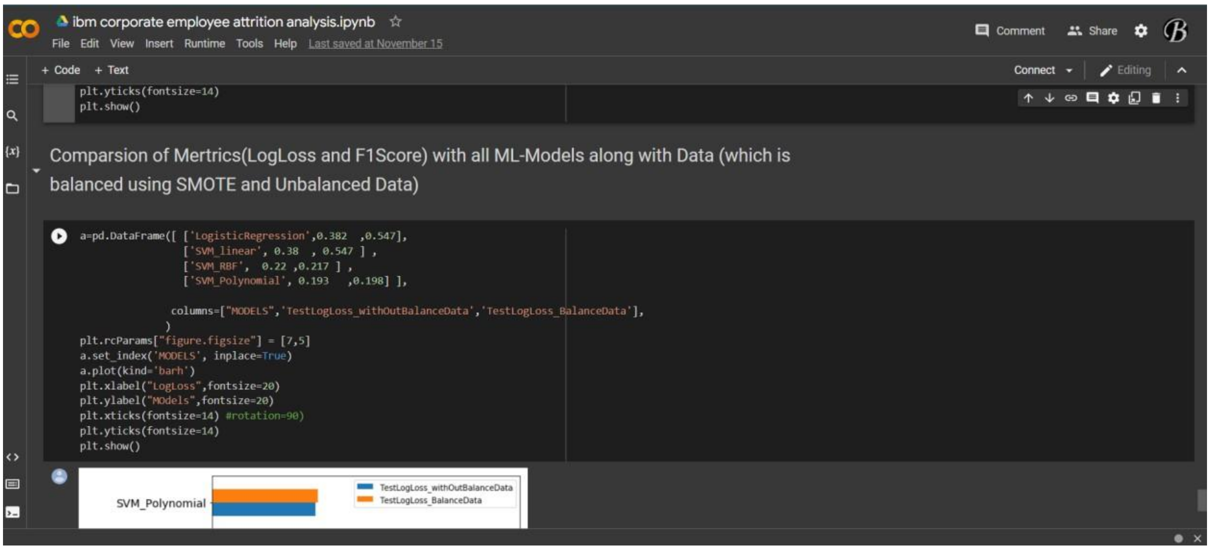


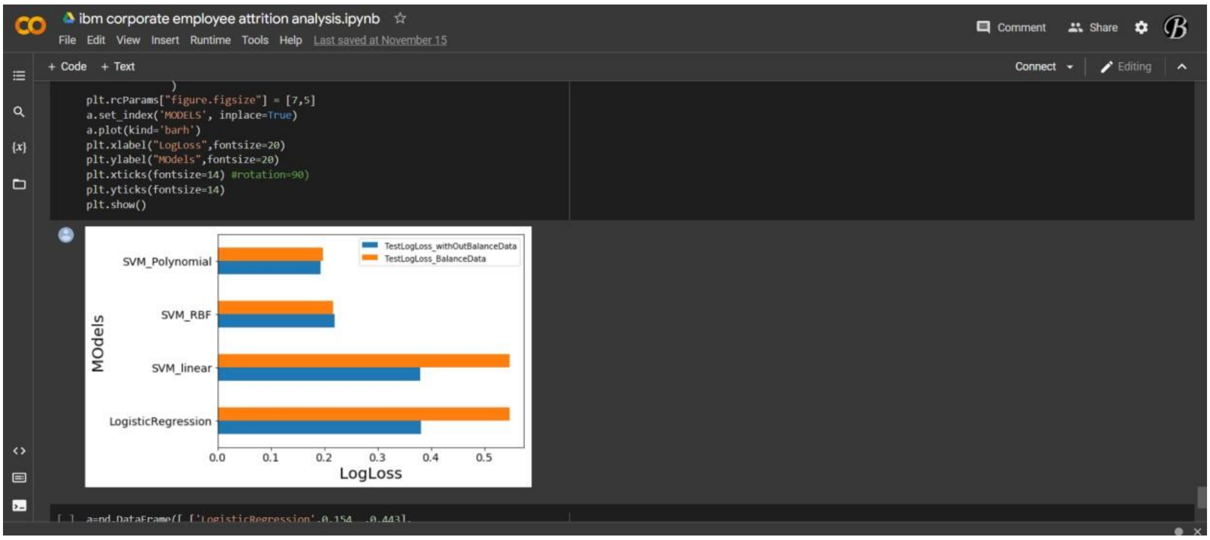
CODING:

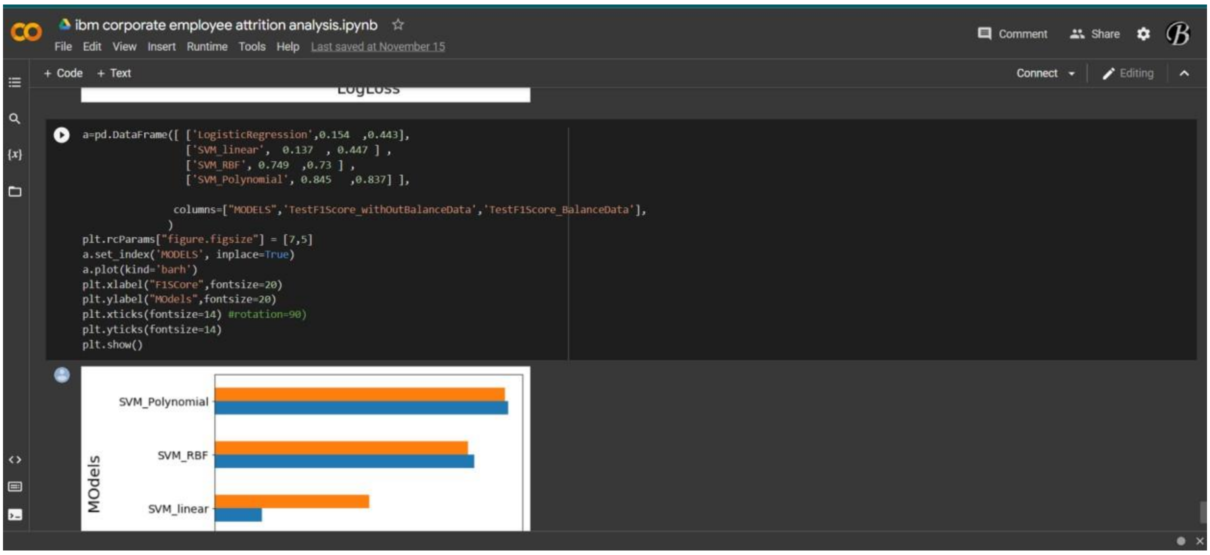
```
import pandas as pd import
matplotlib.pyplot as plt
a=pd.DataFrame([ ['NaiveBayes',0.5192,1.163 ,1.195],
            ['KNN',  0.6288 , 1.006 , 1.083] ,
            ['LR ClassBlance',   0.4406 , 0.986 , 1.006] ,
            ['LR NO ClassBlance', 0.4347 , 1.0114  ,1.0298],
           ['SVM LINEAR',  0.4828 ,  1.0236 , 1.056],
           ['RF with Response coding ', 0.056  , 1.373 , 1.383],
          ['RF with OneHotEncoding', 0.8369 , 1.1908 , 1.1781 ],
           ['SlackModels', 0.5296, 1.124  , 1.19],
           ['max voting', 0.821, 1.167 , 1.196],
          ['LR NO ClassBlance[Count Vectorize UNIGRAM]', 0.6099, 1.157 , 1.11 ],
           ['LR NO ClassBlance[Count Vectorize BIGRAM]', 0.8382, 1.2165 , 1.2227 ]],
          columns=["MODELS",'Train LogLoss','CV LogLoss','Test LogLoss'],
         )
```

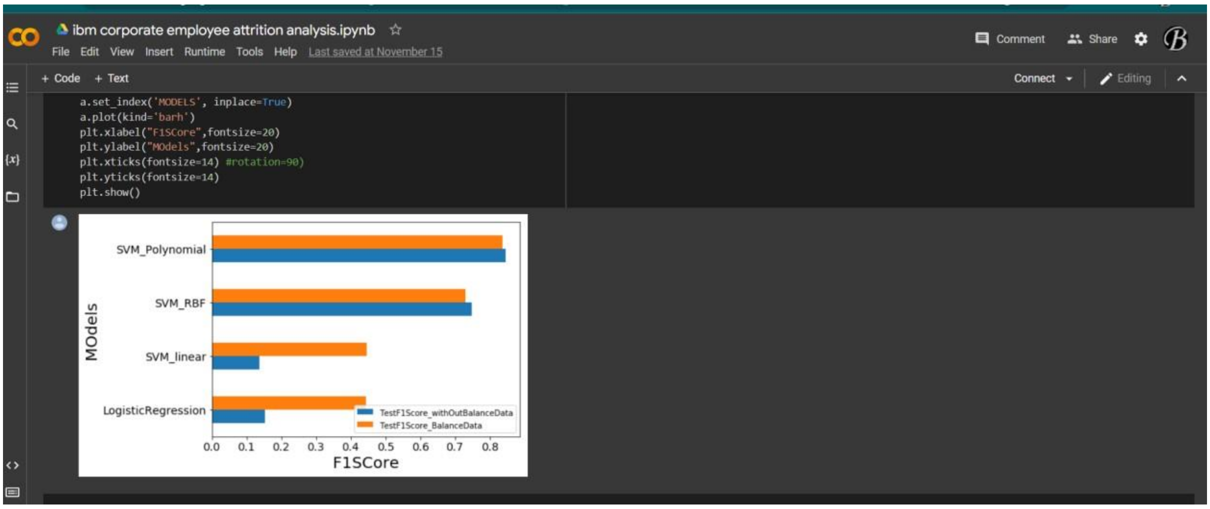
```
#plt.figure(figsize=(1,1))
plt.rcParams["figure.figsize"] = [14,9]
a.set_index('MODELS', inplace=True)
a.plot(kind='barh')
plt.xlabel("LogLoss",fontsize=20)
plt.ylabel("MOdels",fontsize=20)
plt.xticks(fontsize=14)
#rotation=90)
plt.yticks(fontsize=14) plt.show()
```

COMPARSION OF MERTRICS (LOGLOSS AND F1SCORE) WITH ALL ML-MODELS ALONG WITH DATA (WHICH IS BALANCED USING SMOTE AND UNBALANCED DATA)




ibm corporate employee attrition analysis.ipynb
File Edit View Insert Runtime Tools Help Last saved at November 15
Comment
Share
+ Code + Text
Connect
Editing
)
plt.rcParams["figure.figsize"] = [7,5]
a.set_index('MODELS', inplace=True)
a.plot(kind='barh')
plt.xlabel("LogLoss",fontsize=20)
plt.ylabel("MOdels",fontsize=20)
plt.xticks(fontsize=14) #rotation=90)
plt.yticks(fontsize=14)
plt.show()
TestLogLoss_withOutBalanceData
TestLogLoss_BalanceData
SVM_Polynomial
SVM_RBF
SVM_linear
LogisticRegression
MOdels
0.0
0.1
0.2
0.3
0.4
0.5
LogLoss





CODING:

```
a=pd.DataFrame([ ['LogisticRegression',0.382  ,0.547],
            ['SVM_linear', 0.38  , 0.547 ] ,
            ['SVM_RBF',  0.22 ,0.217 ] ,
            ['SVM_Polynomial', 0.193   ,0.198] ],

          columns=["MODELS",'TestLogLoss_withOutBalanceData','TestLogLoss_BalanceData'],
          )
plt.rcParams["figure.figsize"] = [7,5]
```

a.set_index('MODELS', inplace=True)

a.plot(kind='barh')

plt.xlabel("LogLoss",fontsize=20)

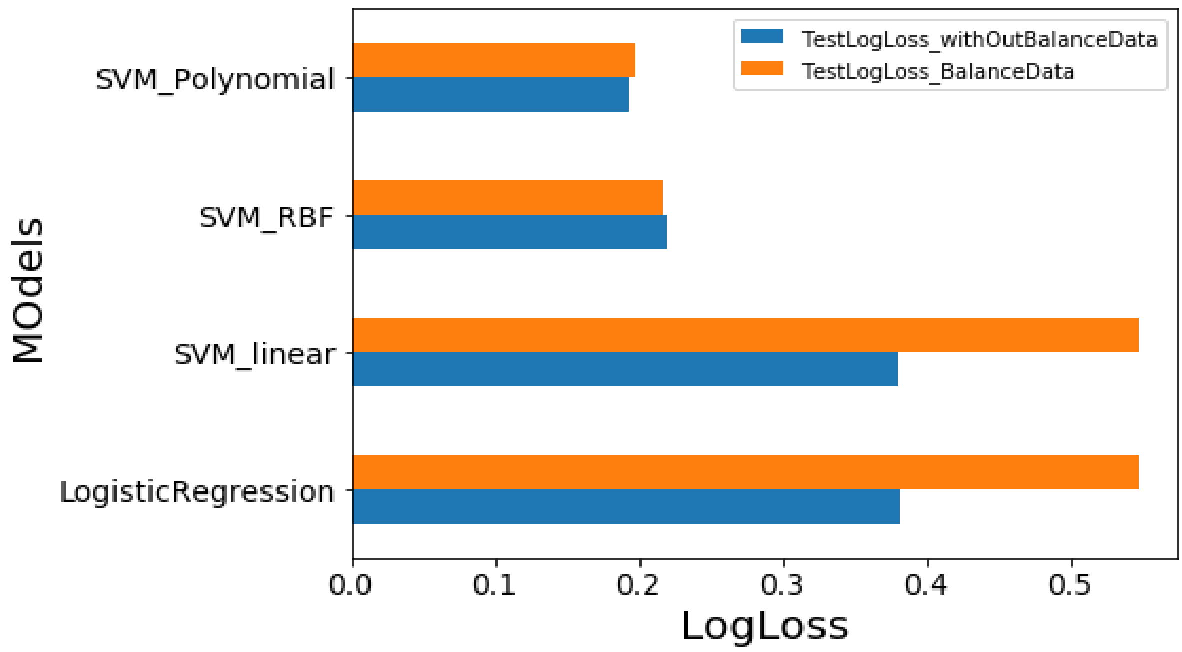
plt.ylabel("MOdels",fontsize=20)

plt.xticks(fontsize=14)

#rotation=90)

plt.yticks(fontsize=14) plt.show()

OUTPUT:



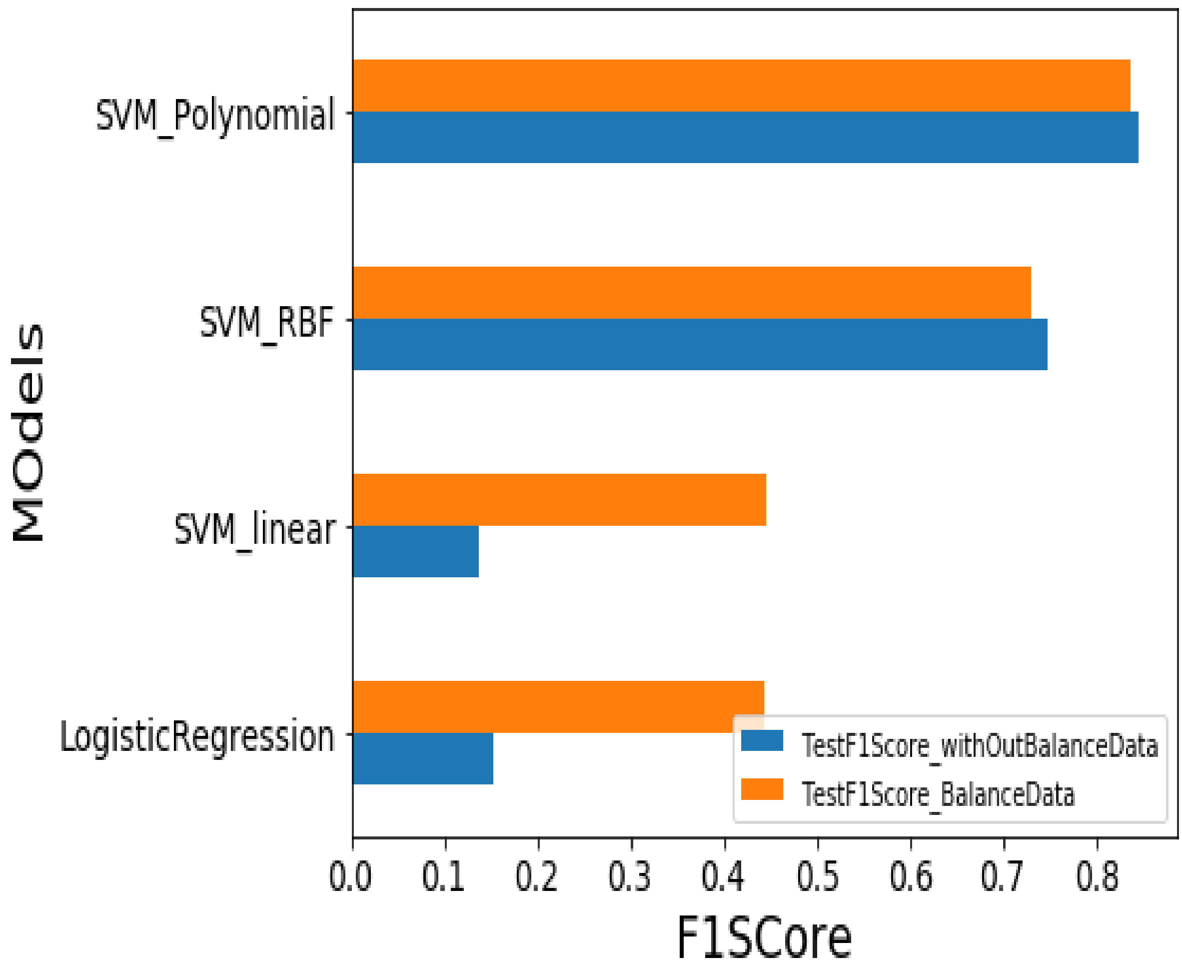
CODING:

a=pd.DataFrame([['LogisticRegression',0.154 ,0.443],

['SVM_linear', 0.137 , 0.447] ,

```
              ['SVM_RBF', 0.749  ,0.73 ] ,
              ['SVM_Polynomial', 0.845   ,0.837] ],

            columns=["MODELS",'TestF1Score_withOutBalanceData','TestF1Score_
BalanceData'],
            )
plt.rcParams["figure.figsize"] = [7,5]
a.set_index('MODELS', inplace=True)
a.plot(kind='barh')
plt.xlabel("F1SCore",fontsize=20)
plt.ylabel("MOdels",fontsize=20)
plt.xticks(fontsize=14)
#rotation=90)
plt.yticks(fontsize=14) plt.show()
```

OUTPUT:



DATA VISUALIZATION CHARTS AND DASHBOARD CREATION

Using the given dataset, we need to create various graphs and charts to highlight the insights and visualizations. For the given problem statment, try to build the following visualizations that suit the solution requirements.

- Employee Attrition by Age
- Attrition by Business Travel

- Attrition by Department, Job Role, Education Level and Marital Status
- Attrition by Salary Hike Percent
- Attrition by No. of Companies Worked
- Attrition by Income Groups
- Attrition by Work Experience Groups
- Dashboard of Attrition of Employees based on Employment details

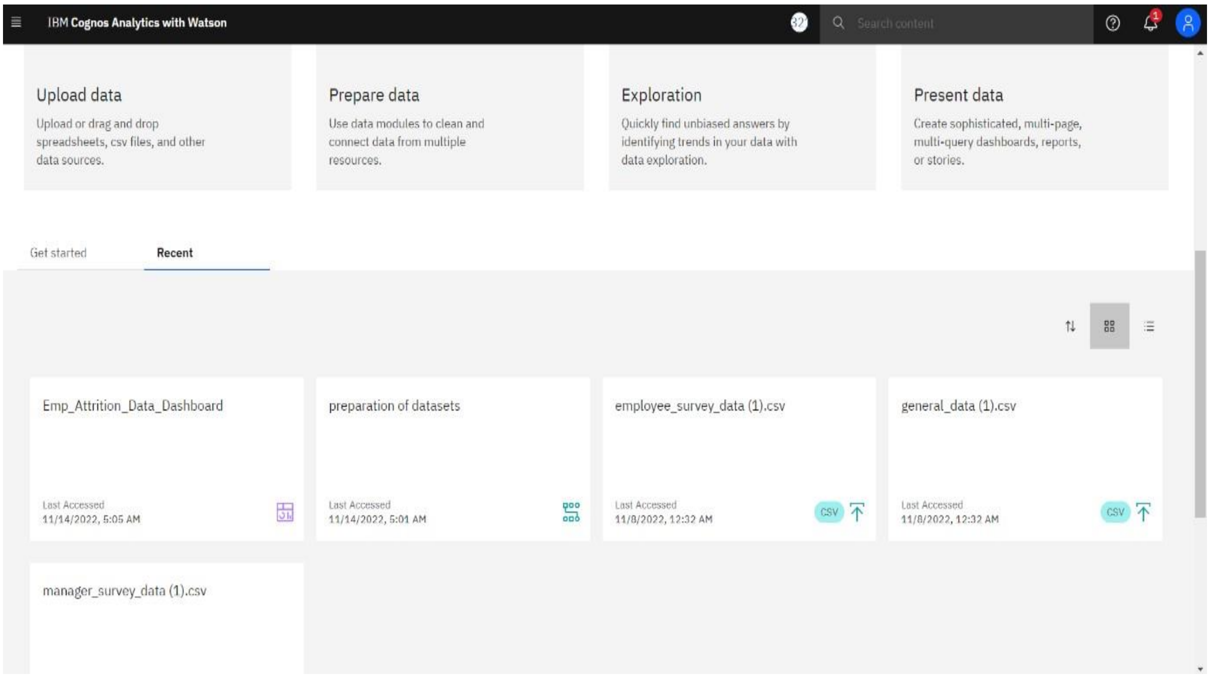
IBM COGNOS ANALYTICS

To create data visualization charts and dashboard we need to login into IBM Cognos analytics. IBM® Cognos® Analytics integrates reporting, modeling, analysis, dashboards, stories, and event management so that you can understand your organization data, and make effective business decisions.This tool is used to give better understanding about the dataset.

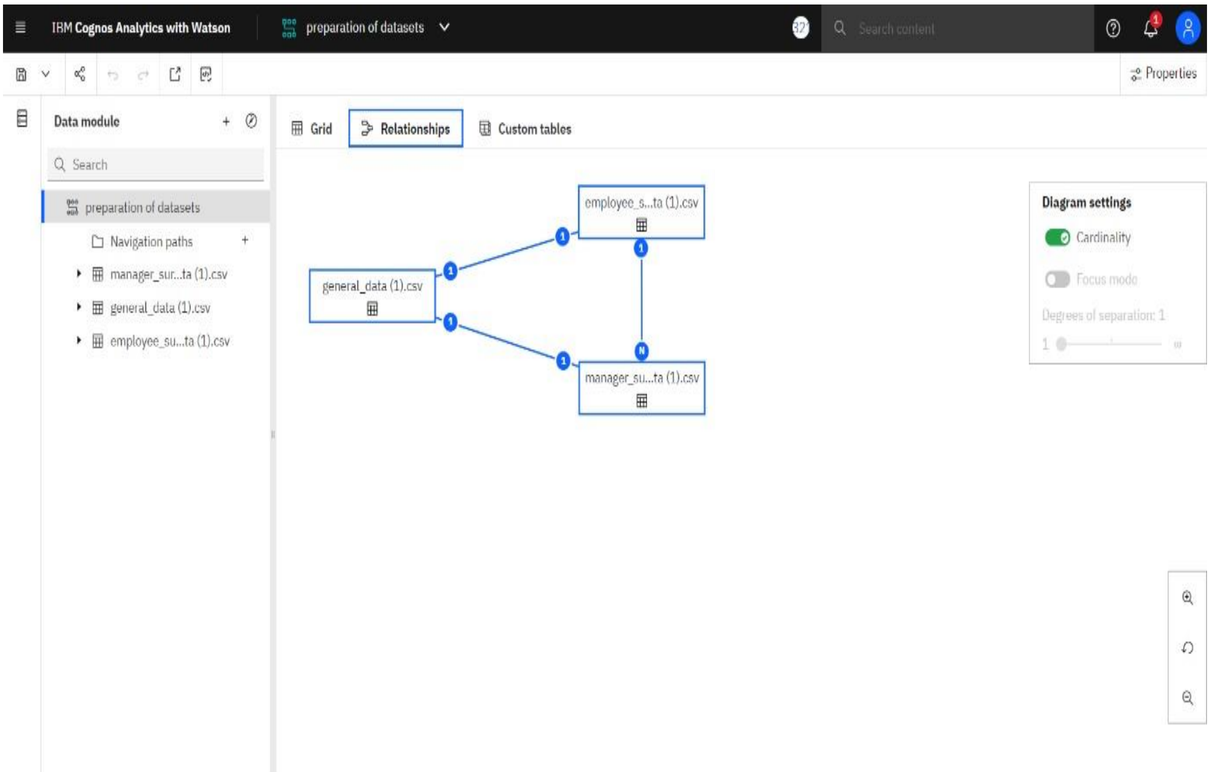
STEPS TO CREATE VISUALIZATION CHARTS AND DASHBOARD CREATION

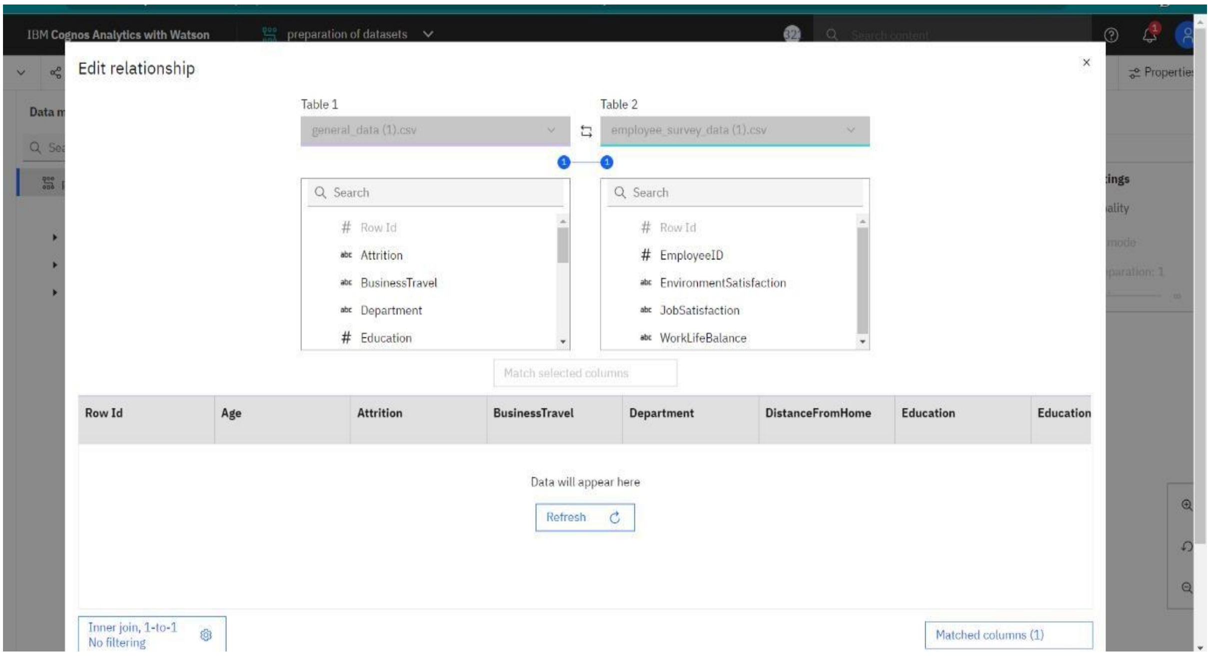
- Uploading of data
- Preparing the data
- Exploration of data
- Creation of Visualization Charts
- Dashboard creation

LOADING THE DATASET:



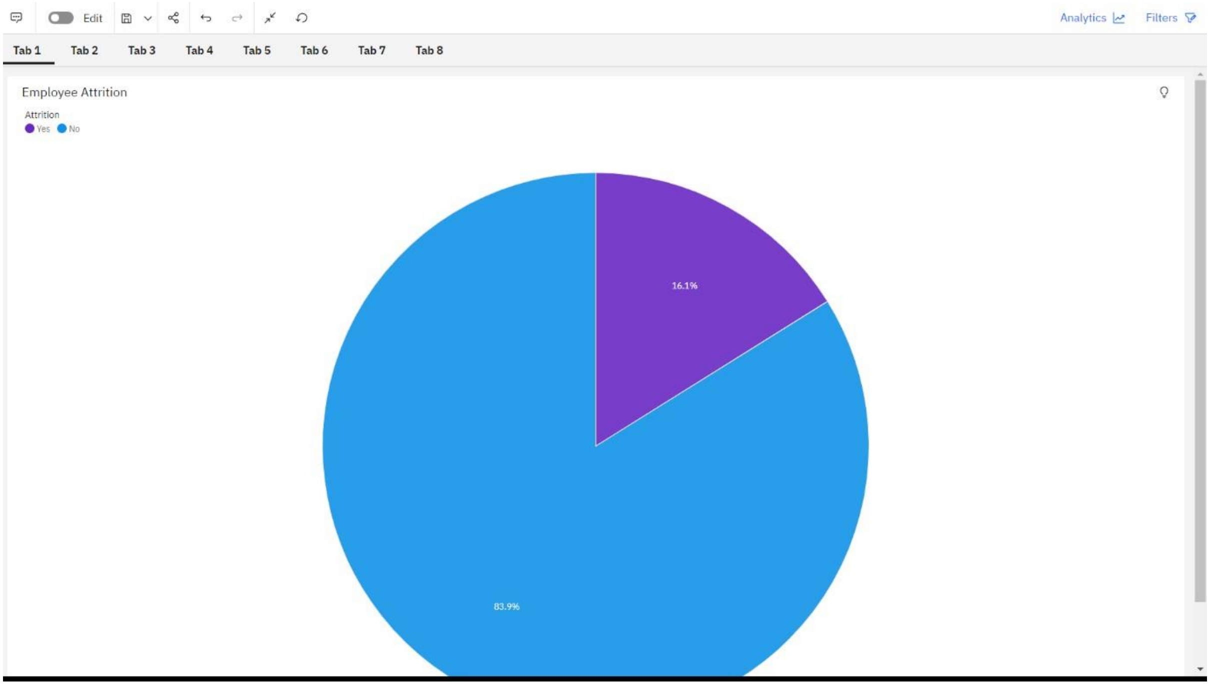
PREPARING THE DATA & EXPLORATION OF DATA





CREATION OF VISUALIZATION CHARTS

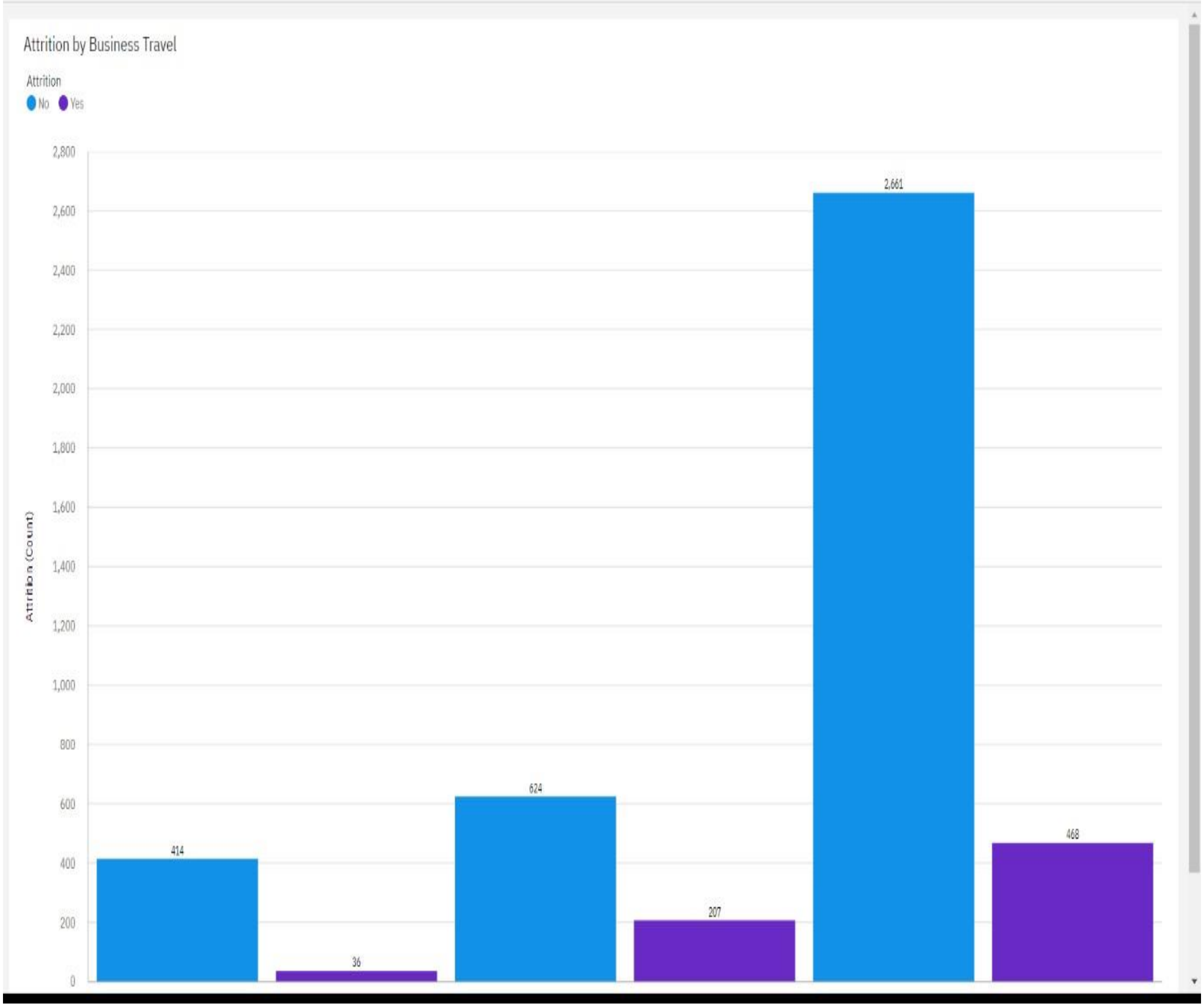
- EMPLOYEE ATTRITION STATUS:



- INFERENCES:

We can understand from the above pie chart that 16.1% of people are willing to leave and 83.3% say no to it

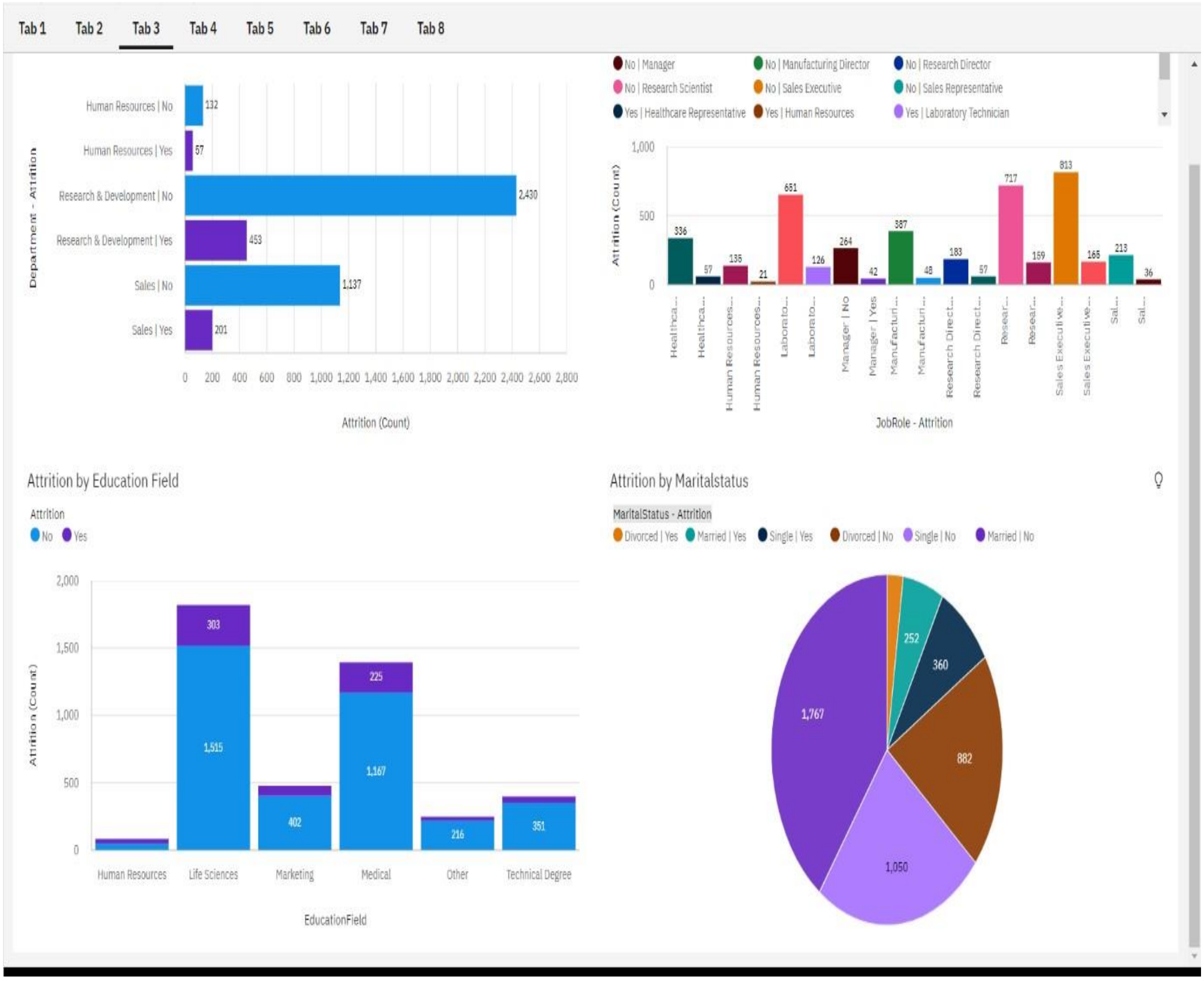
- ATTRITION BY BUSINESS TRAVEL



◦ INFERENCES:

We can understand from the above column chart that 468 people are willing to leave

- ATTRITION BY DEPARTMENT, JOB ROLE, EDUCATION LEVEL AND MARITAL STATUS



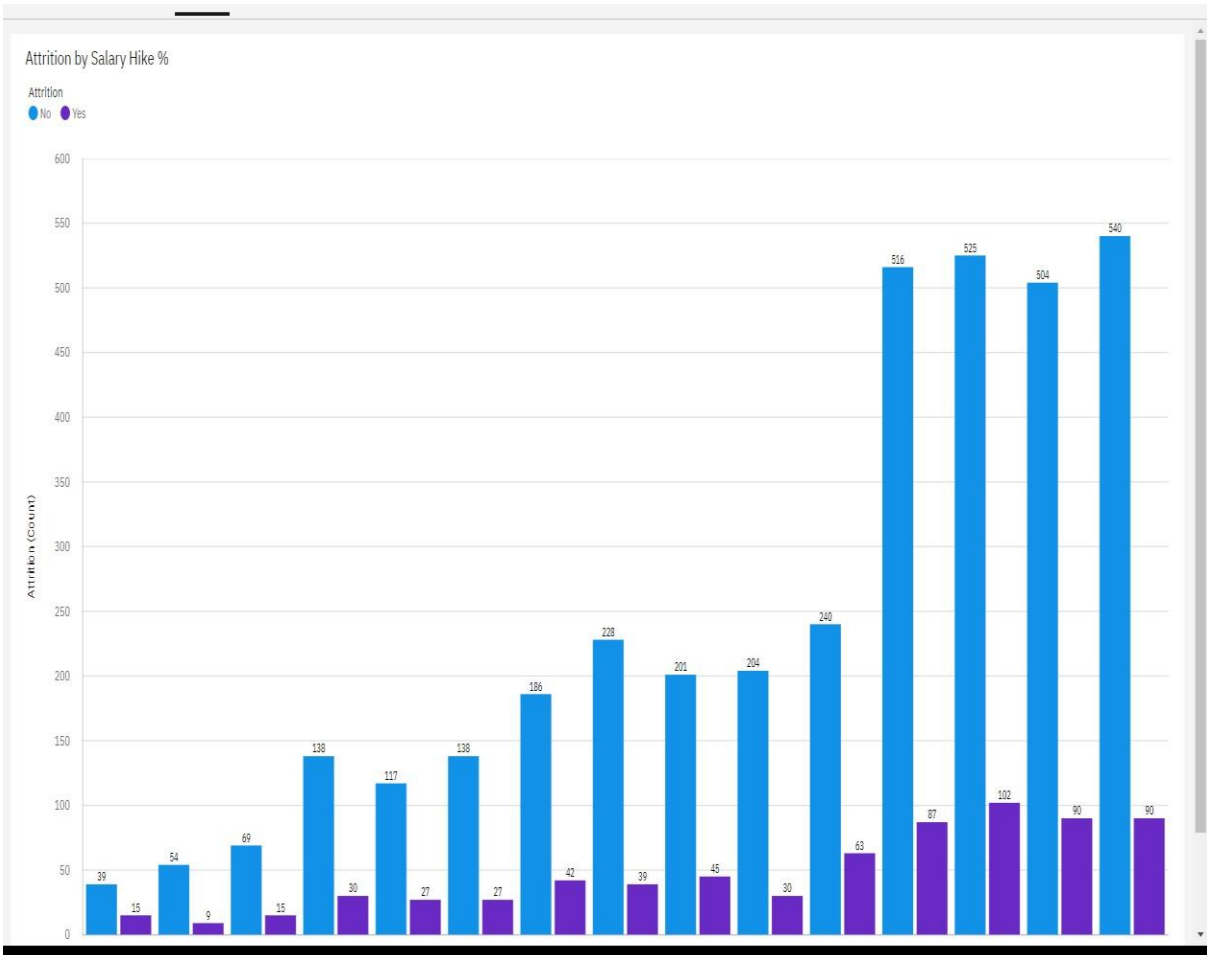
Tab 1
Tab 2
Tab 3
Tab 4
Tab 5
Tab 6
Tab 7
Tab 8
Department - Attrition
Human Resources | No
Human Resources | Yes
Research & Development | No
Research & Development | Yes
Sales | No
Sales | Yes
132
57
2,430
453
1,137
201
0 200 400 600 800 1,000 1,200 1,400 1,600 1,800 2,000 2,200 2,400 2,600 2,800
Attrition (Count)
No | Manager
No | Manufacturing Director
No | Research Director
No | Research Scientist
No | Sales Executive
No | Sales Representative
Yes | Healthcare Representative
Yes | Human Resources
Yes | Laboratory Technician
1,000
500
0
Attrition (Count)
336
57
136
21
651
126
264
42
387
48
183
57
717
159
813
165
213
36
Healthca...
Healthca...
Human Resources...
Human Resources...
Laborato...
Laborato...
Manager | No
Manager | Yes
Manufacturi...
Manufacturi...
Research Direct...
Research Direct...
Resear...
Resear...
Sales Executive...
Sales Executive...
Sal...
Sal...
JobRole - Attrition
Attrition by Education Field
Attrition
No
Yes
2,000
1,500
1,000
500
0
Attrition (Count)
303
1,515
225
1,167
402
216
351
Human Resources
Life Sciences
Marketing
Medical
Other
Technical Degree
EducationField
Attrition by Maritalstatus
MaritalStatus - Attrition
Divorced | Yes
Married | Yes
Single | Yes
Divorced | No
Single | No
Married | No
252
360
1,767
882
1,050

◦

INFERENCES:

We can understand from the above 4 division charts , people are willing to leave % is higher.

- ## ATTRITION BY SALARY HIKE PERCENT



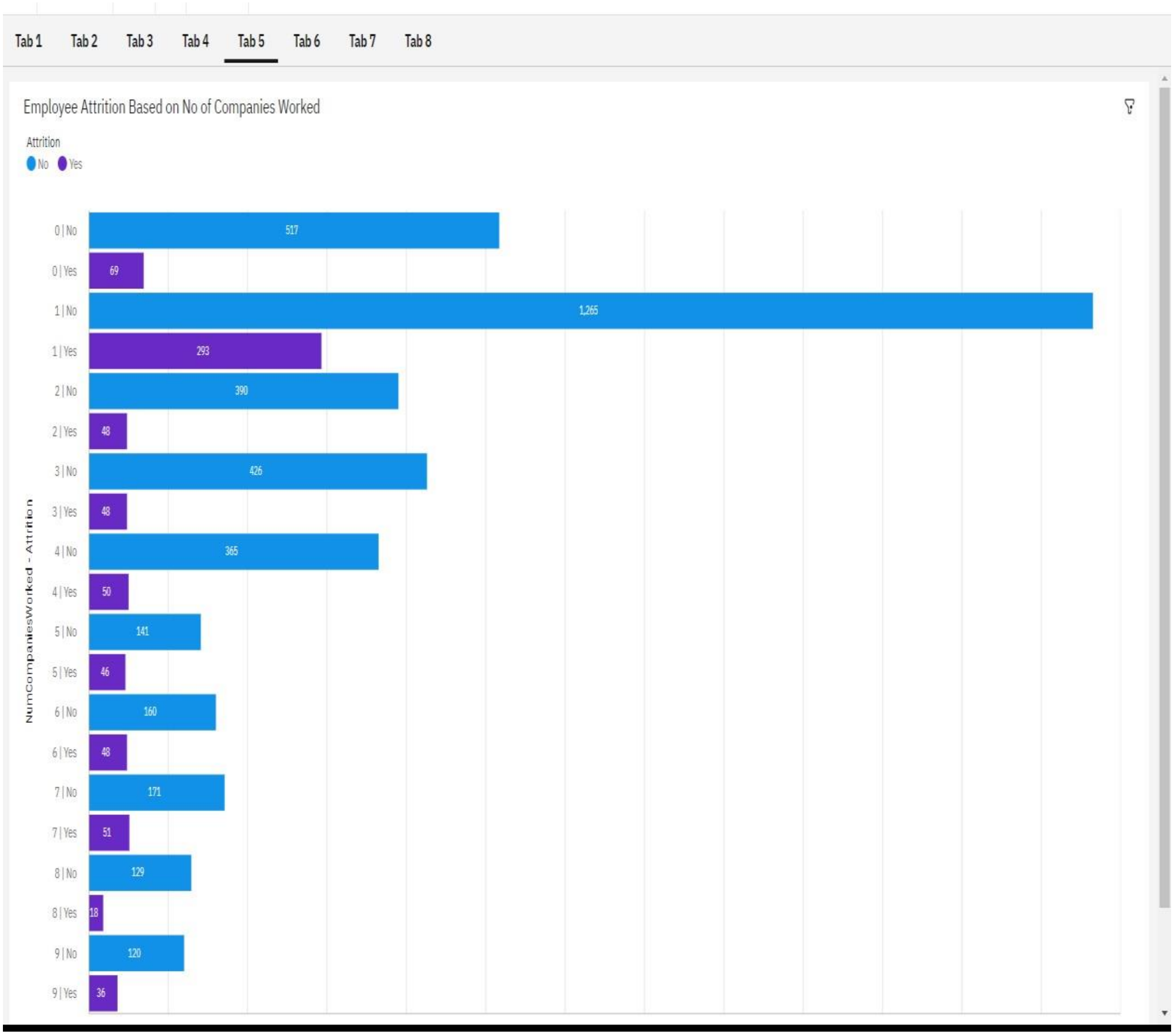
-

INFERENCE:

We can understand from the above charts that % of people leaving and staying.

- ## ATTRITION BY NO. OF COMPANIES WORKED

○

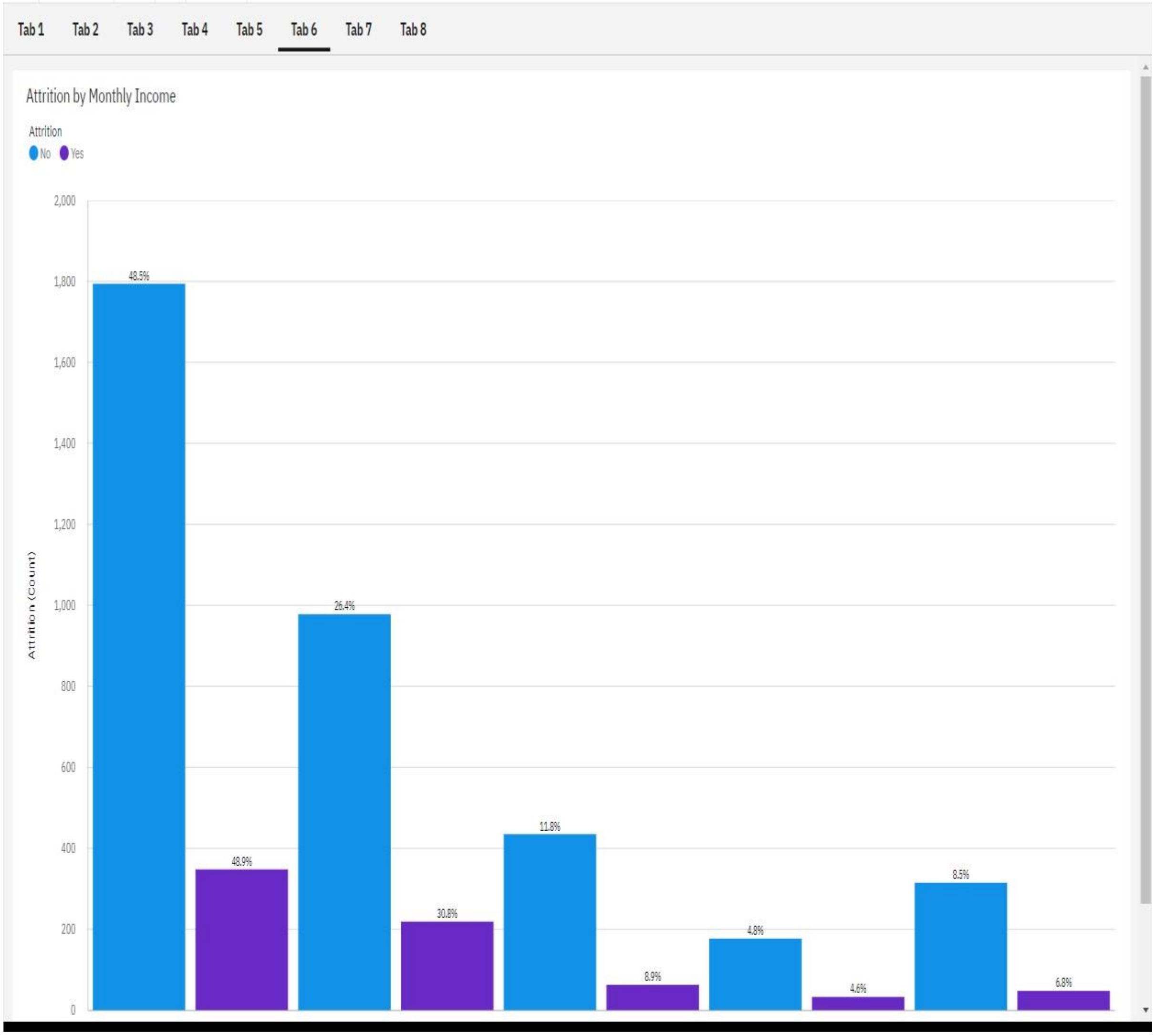


INFERENCES:

We can understand from the above charts that % of people leaving and staying.

◦

- ATTRITION BY INCOME GROUPS

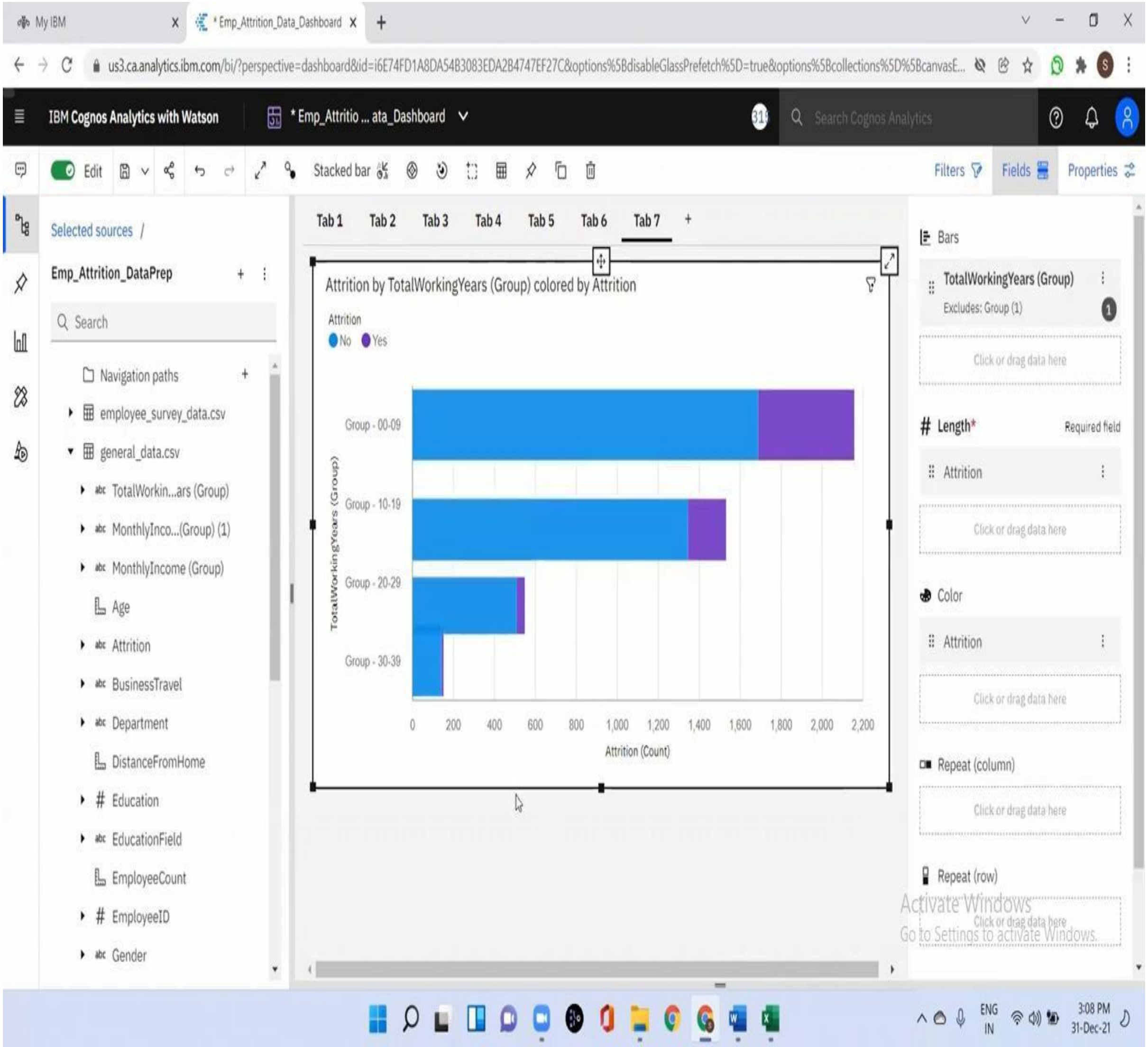


INFERENCE:

○

We can understand from the above charts that % of people leaving and staying.

- ATTRITION BY WORK EXPERIENCE GROUPS

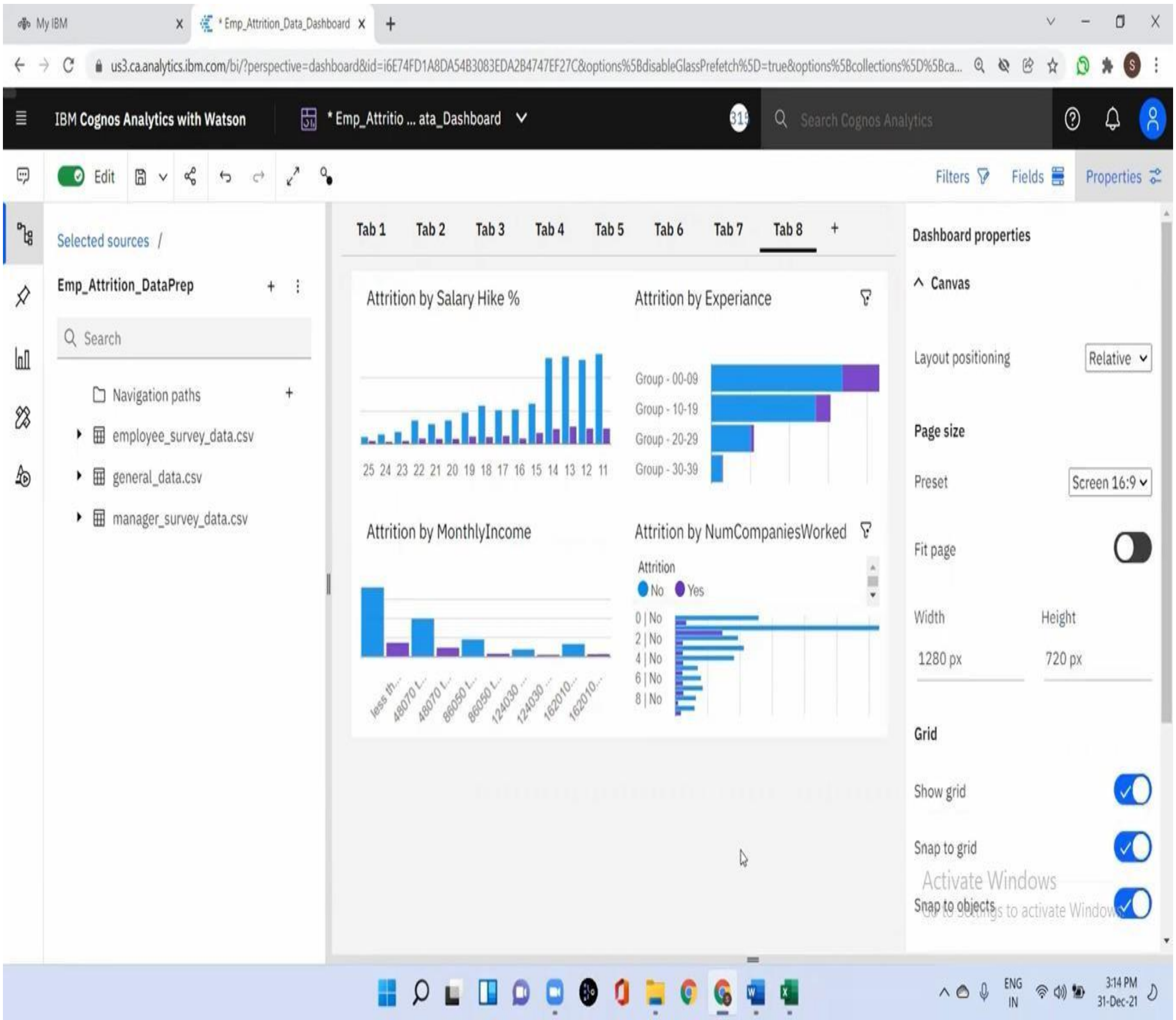


○ INFERENCES:

。

We can understand from the above charts that % of people leaving and staying.

- DASHBOARD OF ATTRITION OF EMPLOYEES BASED ON EMPLOYMENT DETAILS



- INFERENCES

We can understand from the above charts that % of people leaving and staying.

FINDING OF THIS PROJECT:

A total of 24 variables, collected from 4 sources were used to predict the probability of an employee leaving the company in the next year, using a logistic regression model

- Logistic Regression Model* is able to correctly identify 77% of employees that were likely to churn
- It is also able to identify employees that are not likely to churn, with 77% accuracy

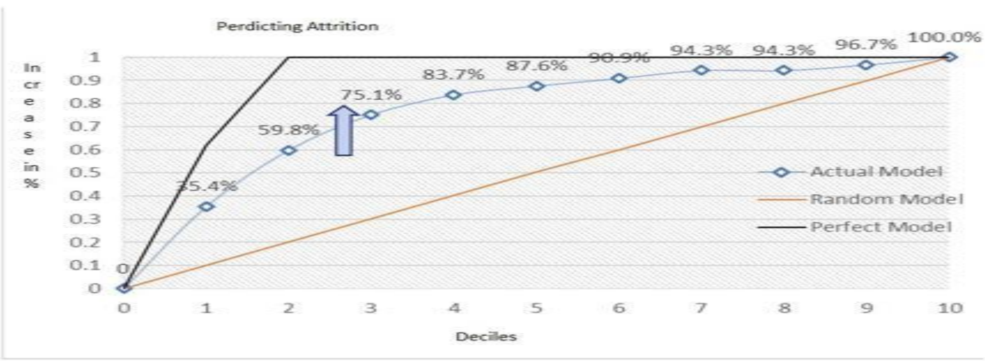
➤ KS STATISTIC FALLS IN 3RD DECILE (TOP 30%)

- Hence, it would be beneficial to target 30% of your employees most likely to leave, and work on making them stay.
- Targeting fewer employees (top 20% or top 10%) will not identify enough employees likely to leave

Targeting more employees (top 40% or top 50%) will be inefficient

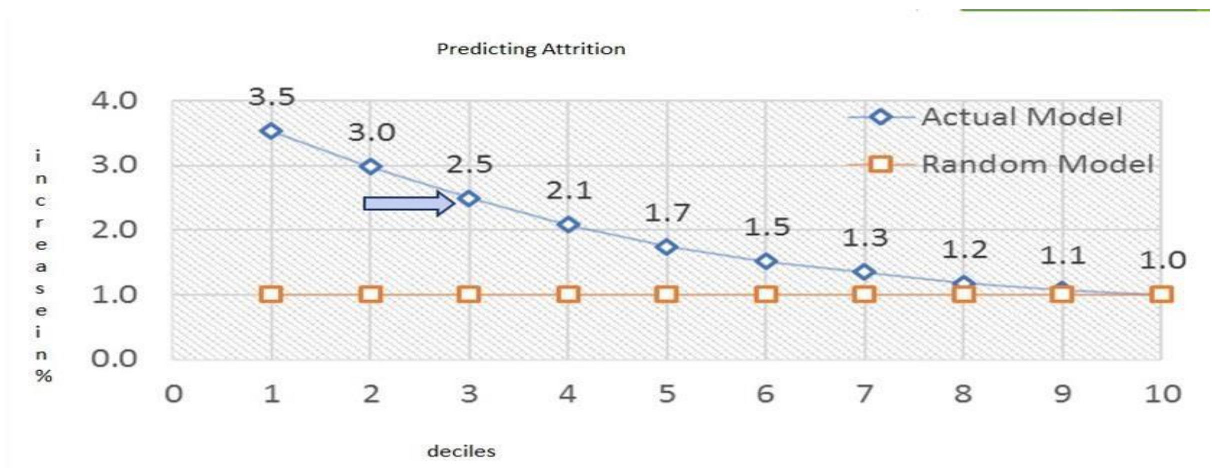
➤ Predicting Attrition – Model is able to capture 75% of employees likely to leave

- Model is able to identify 75% of the employees likely to leave in the first 3 deciles



➤ Predicting Attrition – Model performs 2.5 times better than a random reach out

- Using the model offers a "lift" of 2.5 for the 3rd decile



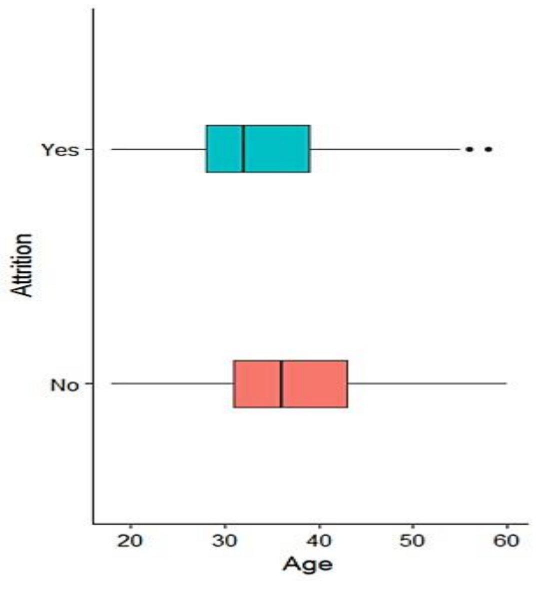
RECOMMENDATIONS – WHAT FACTORS MAKE EMPLOYEES STAY/LEAVE? (1/4)

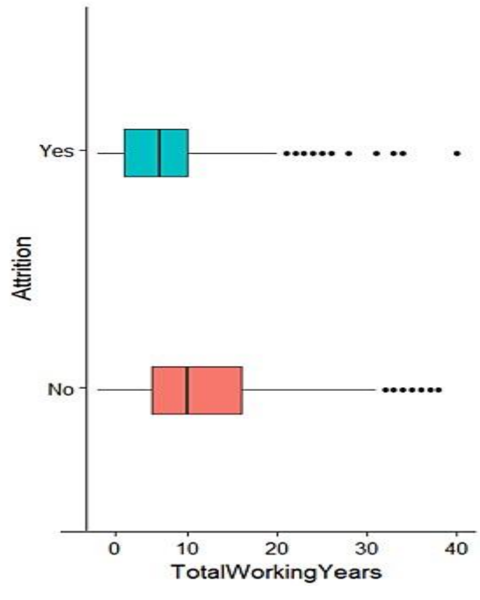
➤ EXPERIENCE

- Employees that have worked for a total of 7 years or less are more likely to leave*
- Employees that have worked for a total of 10 years or more are more likely to stay*

➤ AGE

- Employees aged 36 years and above are more likely to stay*
- Employees aged 32 years and below are more likely to leave*





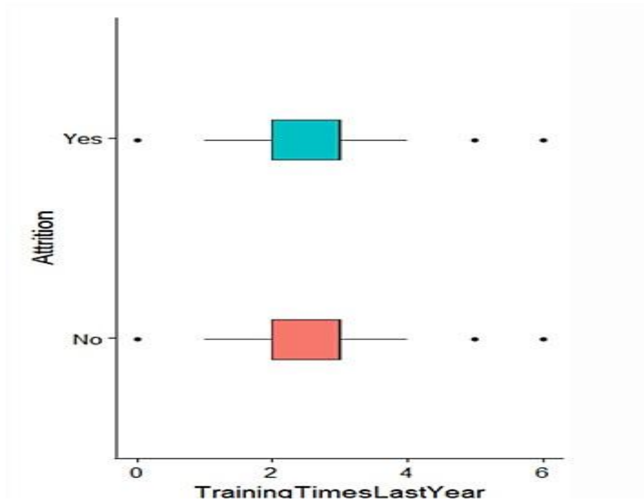
RECOMMENDATIONS – WHAT FACTORS MAKE EMPLOYEES STAY/LEAVE? (2/4)

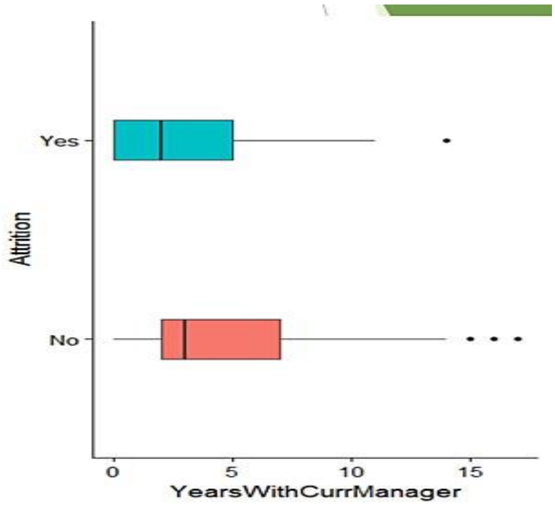
➤ TRAINING

- Employees that got 3 or more training sessions last year are more likely to stay*
- Employees that got 2 or fewer training sessions last year are more likely to leave*

➤ YEARS WITH CURRENT MANAGER

- Employees that have spent 3 years or more under the same manager are more likely to stay*
- Employees that have spent 2 years or less under the same manager are more likely to leave*





RECOMMENDATIONS – WHAT FACTORS MAKE EMPLOYEES STAY/LEAVE? (3/4)

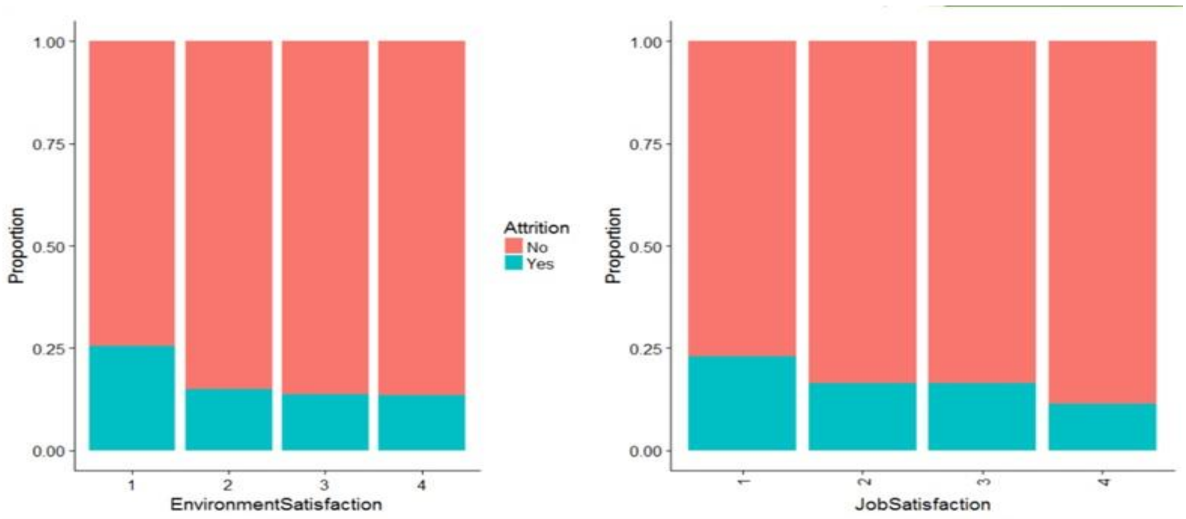
➤ JOB SATISFACTION

- Employees that have medium, high or very high levels of job satisfaction, are more likely to stay*
- Employees that have low levels of job satisfaction, are more likely to leave*

➤ ENVIRONMENT SATISFACTION

•Employees that have medium, high or very high levels of environment satisfaction, are more likely to stay*

•Employees that have low levels of environment satisfaction, are more likely to leave*



- RECOMMENDATIONS – WHAT FACTORS MAKE EMPLOYEES STAY/LEAVE? (4/4)

➤ AVERAGE WORK HOURS

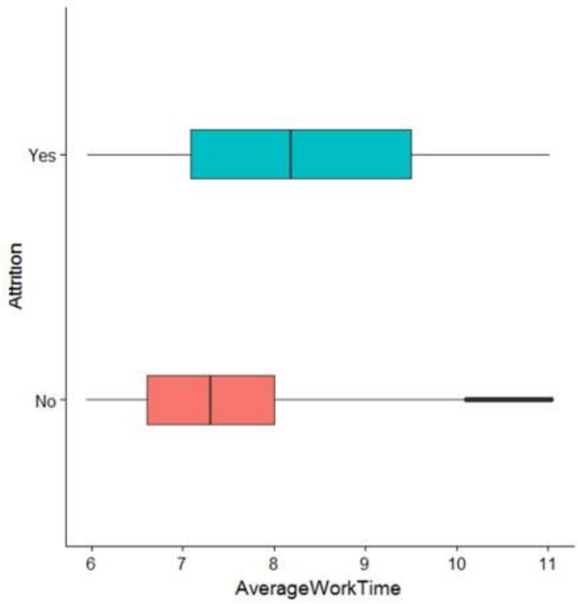
•Employees that, on average work for 7.3 hours or less, are more likely to stay*

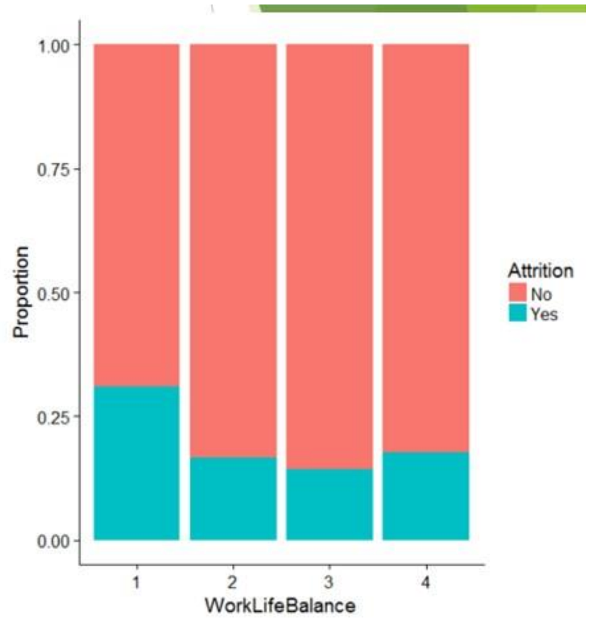
•Employees that, on average work for 8.2 hours or more, are more likely to leave*

➤ WORK LIFE BALANCE

•Employees that rated their work life balance as good, better or best, are more likely to stay**

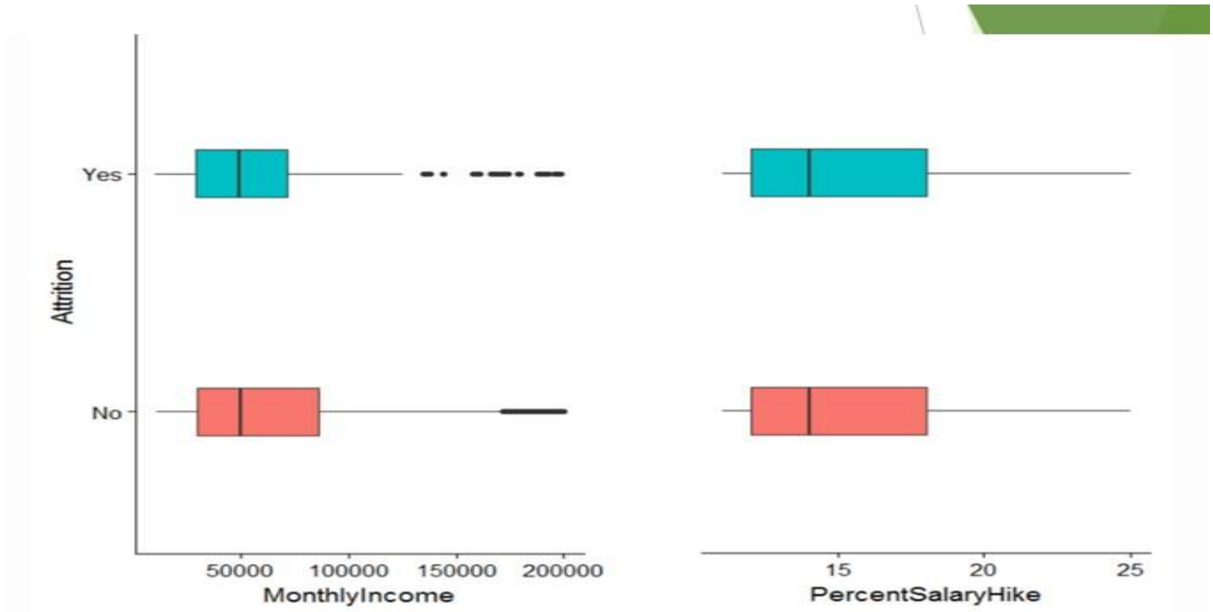
•Employees that rated their work life balance as bad, are more likely to leave**





• RECOMMENDATIONS – FACTORS THAT SURPRISINGLY DON'T AFFECT ATTRITION

❖ Monthly Income and Percent Salary Hike do not affect attrition*



RECOMMENDATIONS

- CURRENT EMPLOYEES:

- •Work life balance should be improved
- •Work environment should be improved
- •The manager of an employee should not be changed very often

- Employees should be provided relevant training regularly, especially for its younger employees

- FUTURE EMPLOYEES (CHANGES IN HIRING PROCESS):

  - The company should follow either one of the strategies given below –
    - Hire older people with decent work experience
    - Hire young people and train them appropriately
  - It could also opt for a combination of the two