

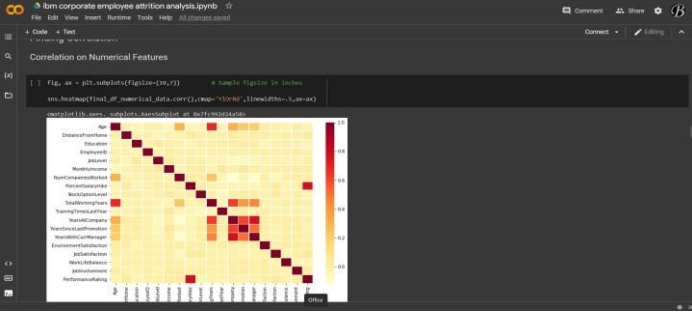
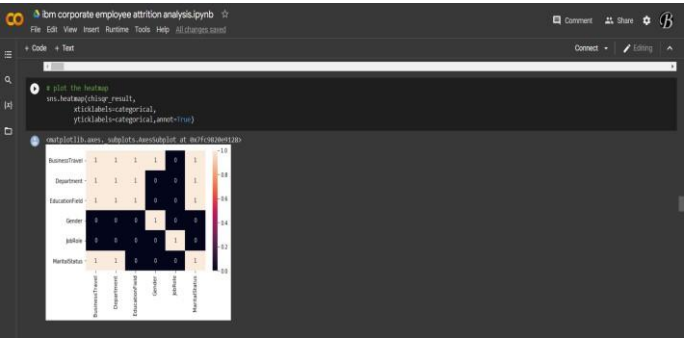
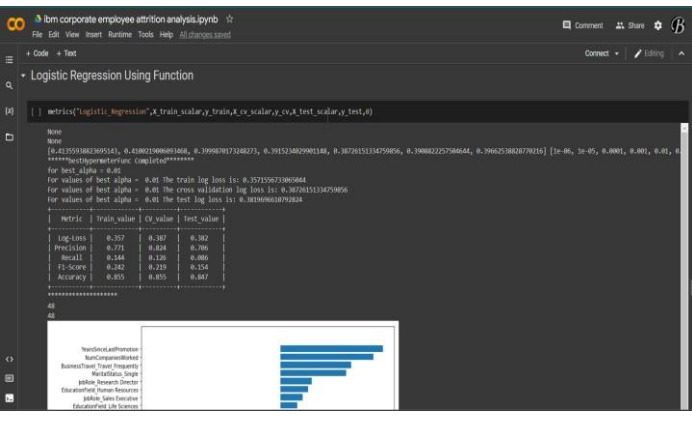
Project Development Phase
Model Performance Test

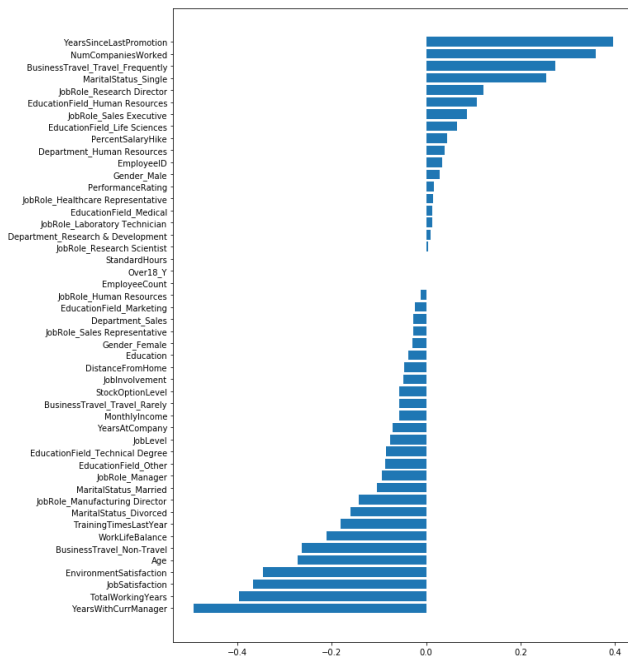
Date	10 November 2022
Team ID	PNT2022TMID06047
Project Name	Project - Corporate Employee Attrition Analytics
Maximum Marks	10 Marks

Model Performance Testing:

Project team shall fill the following information in model performance testing template.

S.No	Parameter	Values	Screenshot
.	r		

1.	Metrics	<p>Regression Model: R2 score -79%</p> <p>Classification Model: Confusion Matrix - 79% Accuracy Score- 76%</p> <p>& Classification Report –</p> <p>1.Correlation 2.Confusion matrix 3.Logistic regression 4.Linear SVM 5.RBF Kernel 6.Poly Kernel</p>	  
----	---------	---	--



```

IBM corporate employee attrition analysis.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Test
Connect + Editing

SVM
Using Linear Kernel

In [ ]:
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error, accuracy_score, confusion_matrix, classification_report

# Train and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Create the SVM model with Linear Kernel
svm = SVC(kernel='linear')

# Train the model
svm.fit(X_train, y_train)

# Predict on test data
y_pred = svm.predict(X_test)

# Evaluate the model
print("R2 Score: ", r2_score(y_test, y_pred))
print("Mean Squared Error: ", mean_squared_error(y_test, y_pred))
print("Mean Absolute Error: ", mean_absolute_error(y_test, y_pred))
print("Accuracy Score: ", accuracy_score(y_test, y_pred))
print("Confusion Matrix: ")
print(confusion_matrix(y_test, y_pred))
print("Classification Report: ")
print(classification_report(y_test, y_pred))

# Hyperparameter tuning for Linear Kernel
from sklearn.model_selection import GridSearchCV
param_grid = {'C': [0.1, 1, 10, 100, 1000], 'gamma': [0.001, 0.01, 0.1, 1, 10]}
grid_search = GridSearchCV(svm, param_grid, cv=5, scoring='r2')
grid_search.fit(X_train, y_train)

print("Best parameters for Linear Kernel: ", grid_search.best_params_)
print("Best R2 Score for Linear Kernel: ", grid_search.best_score_)

# Hyperparameter tuning for RBF Kernel
param_grid = {'C': [0.1, 1, 10, 100, 1000], 'gamma': [0.001, 0.01, 0.1, 1, 10]}
grid_search = GridSearchCV(svm, param_grid, cv=5, scoring='r2')
grid_search.fit(X_train, y_train)

print("Best parameters for RBF Kernel: ", grid_search.best_params_)
print("Best R2 Score for RBF Kernel: ", grid_search.best_score_)

# Hyperparameter tuning for Poly Kernel
param_grid = {'C': [0.1, 1, 10, 100, 1000], 'gamma': [0.001, 0.01, 0.1, 1, 10], 'degree': [1, 2, 3, 4, 5]}
grid_search = GridSearchCV(svm, param_grid, cv=5, scoring='r2')
grid_search.fit(X_train, y_train)

print("Best parameters for Poly Kernel: ", grid_search.best_params_)
print("Best R2 Score for Poly Kernel: ", grid_search.best_score_)

```

```

IBM corporate employee attrition analysis.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Test
Connect + Editing

SVM
Using RBF Kernel

In [ ]:
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error, accuracy_score, confusion_matrix, classification_report

# Train and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Create the SVM model with RBF Kernel
svm = SVC(kernel='rbf')

# Train the model
svm.fit(X_train, y_train)

# Predict on test data
y_pred = svm.predict(X_test)

# Evaluate the model
print("R2 Score: ", r2_score(y_test, y_pred))
print("Mean Squared Error: ", mean_squared_error(y_test, y_pred))
print("Mean Absolute Error: ", mean_absolute_error(y_test, y_pred))
print("Accuracy Score: ", accuracy_score(y_test, y_pred))
print("Confusion Matrix: ")
print(confusion_matrix(y_test, y_pred))
print("Classification Report: ")
print(classification_report(y_test, y_pred))

# Hyperparameter tuning for RBF Kernel
from sklearn.model_selection import GridSearchCV
param_grid = {'C': [0.1, 1, 10, 100, 1000], 'gamma': [0.001, 0.01, 0.1, 1, 10]}
grid_search = GridSearchCV(svm, param_grid, cv=5, scoring='r2')
grid_search.fit(X_train, y_train)

print("Best parameters for RBF Kernel: ", grid_search.best_params_)
print("Best R2 Score for RBF Kernel: ", grid_search.best_score_)

# Hyperparameter tuning for Poly Kernel
param_grid = {'C': [0.1, 1, 10, 100, 1000], 'gamma': [0.001, 0.01, 0.1, 1, 10], 'degree': [1, 2, 3, 4, 5]}
grid_search = GridSearchCV(svm, param_grid, cv=5, scoring='r2')
grid_search.fit(X_train, y_train)

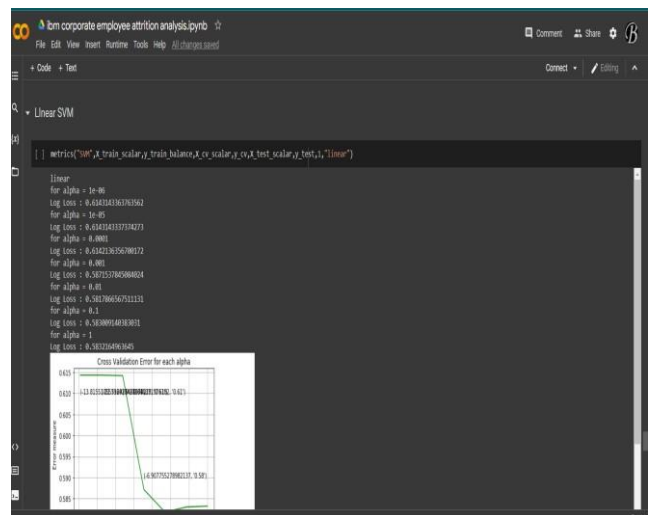
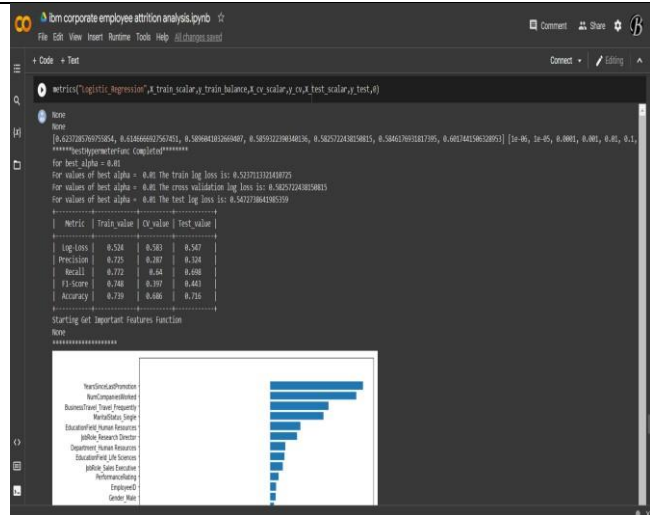
print("Best parameters for Poly Kernel: ", grid_search.best_params_)
print("Best R2 Score for Poly Kernel: ", grid_search.best_score_)

Starting get Important features function
Kernel is rbf so, we cannot get the important features using coef. function

```

--	--	--	--

			<div><div>IBM corporate employee attrition analysis.ipynb</div><div><div>File Edit View Insert Runtime Tools Help All changes saved</div><div><div>Code + Text</div><div>Kernel is not running</div><div>Kernel is not running, we cannot get the important features using coef_ function</div></div><div>Using Poly Kernel</div><div><pre># Importing the dataset X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0) # Feature Scaling from sklearn.preprocessing import StandardScaler sc = StandardScaler() X_train = sc.fit_transform(X_train) X_test = sc.transform(X_test) # Fitting the Logistic Regression model to the training set logit = LogisticRegression() logit.fit(X_train, y_train) # Making predictions on the test set y_pred = logit.predict(X_test) # Evaluating the model from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score cm = confusion_matrix(y_test, y_pred) acc = accuracy_score(y_test, y_pred) pre = precision_score(y_test, y_pred) rec = recall_score(y_test, y_pred) f1 = f1_score(y_test, y_pred) print('Confusion Matrix: \n', cm) print('Accuracy: \n', acc) print('Precision: \n', pre) print('Recall: \n', rec) print('F1 Score: \n', f1)</pre></div><div>Starting the important features function</div><div>Kernel is not running, we cannot get the important features using coef_ function</div></div></div> <div><div>IBM corporate employee attrition analysis.ipynb</div><div><div>File Edit View Insert Runtime Tools Help All changes saved</div><div><div>Code + Text</div><div>Kernel is not running</div><div>Kernel is not running, we cannot get the important features using coef_ function</div></div><div>With Balanced Data : For Balancing Data using SMOTE function from Imblearn Package</div><div><pre>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': {}".format(sum(y_train==0))) sm = SMOTE(random_state=2) X_train_balanced, y_train_balanced = sm.fit_sample(X_train, y_train.ravel()) print("After oversampling, the shape of train X: {}".format(X_train_balanced.shape)) print("After oversampling, the shape of train y: {}".format(y_train_balanced.shape)) print("After oversampling, counts of Label '1': {}".format(sum(y_train_balanced==1))) print("After oversampling, counts of Label '0': {}".format(sum(y_train_balanced==0))) # Before oversampling, counts of Label '1': 405 # Before oversampling, counts of Label '0': 2307 # After oversampling, the shape of train X: (4014, 40) # After oversampling, the shape of train y: (4014,) # After oversampling, counts of Label '1': 2307 # After oversampling, counts of Label '0': 2307 # Feature Scaling scaler = StandardScaler() scaler.fit(X_train_balanced) X_train_balanced = scaler.transform(X_train_balanced) X_test_balanced = scaler.transform(X_test)</pre></div></div></div>
--	--	--	--



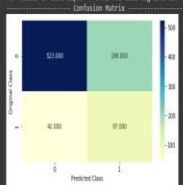
IBM corporate employee attrition analysis.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Code + Test

Linear SVM

```
metrics("SVM",X_train_scaled,y_train_balance,X_cv_scaled,y_cv,X_test_scaled,y_test,1,"linear")
```



Linear

[0.5318181818181818, 0.5318181818181818, 0.5318181818181818, 0.5318181818181818, 0.5318181818181818, 0.5318181818181818, 0.5318181818181818, 0.5318181818181818, 0.5318181818181818, 0.5318181818181818] [1e-05, 1e-05, 0.0001, 0.0001, 0.01, 0.01, 0.1, 1]

*****testwatermar: Completed*****

for best alpha = 0.01

for values of best alpha = 0.01 the train log loss is: 0.52786229624395

for values of best alpha = 0.01 the cross validation log loss is: 0.53178656751131

for values of best alpha = 0.01 the test log loss is: 0.53486276986466

Confusion Matrix


IBM corporate employee attrition analysis.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Code + Test

Linear SVM

```
metrics("SVM",X_train_scaled,y_train_balance,X_cv_scaled,y_cv,X_test_scaled,y_test,1,"linear")
```



Linear SVM

Metric	Train value	Cv value	Test value
log-loss	0.528	0.562	0.547
Precision	0.732	0.699	0.709
Recall	0.776	0.622	0.698
F1 Score	0.752	0.664	0.703
Accuracy	0.745	0.692	0.722


Starting Get Important Features Function

IBM corporate employee attrition analysis.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Code + Test

Log loss : 0.4751615193748624
for alpha = 1
Log loss : 0.7210721072107211



Cross Validation Error for each alpha

Log Loss

Alpha

[0.5318181818181818, 0.5318181818181818, 0.5318181818181818, 0.5318181818181818, 0.5318181818181818, 0.5318181818181818, 0.5318181818181818, 0.5318181818181818, 0.5318181818181818, 0.5318181818181818] [1e-05, 1e-05, 0.0001, 0.0001, 0.01, 0.01, 0.1, 1]

*****testwatermar: Completed*****

for best alpha = 1

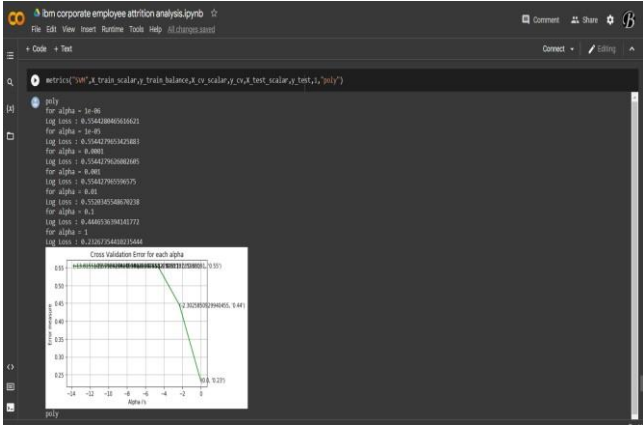
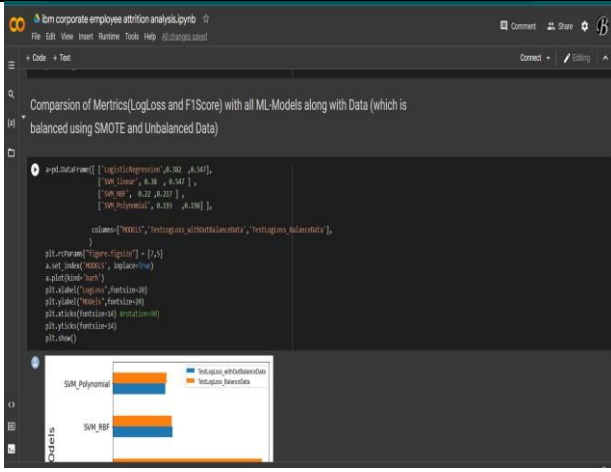
for values of best alpha = 1 the train log loss is: 0.467075259063102

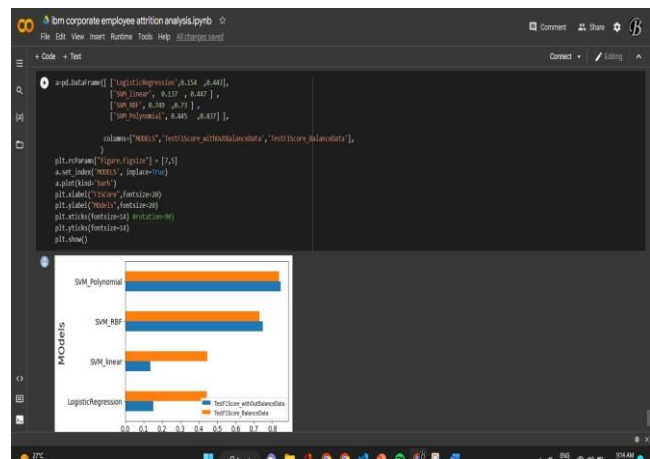
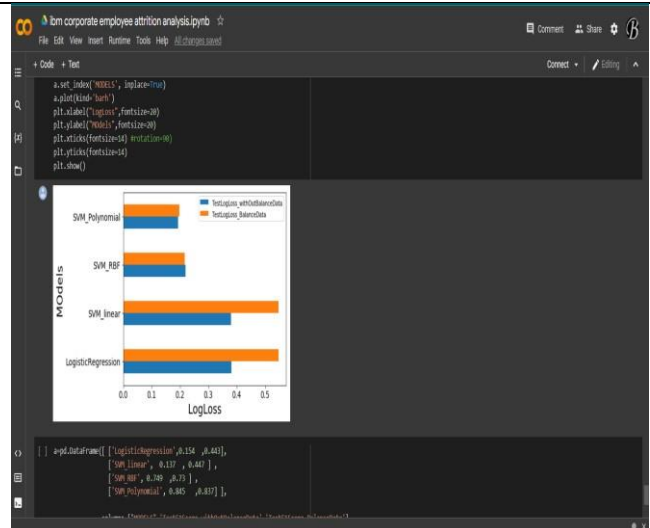
for values of best alpha = 1 the cross validation log loss is: 0.7210721072107211

for values of best alpha = 1 the test log loss is: 0.7210721072107211

Confusion Matrix

--	--	--	--

			
2.	Tune the Model	<p>Hyperparameter Tuning – 80%</p> <p>Validation Method –</p> <ol style="list-style-type: none"> 1. predictive analysis 2. svm model 3. poly kernel 4. RBF Kernel 5. regression analysis 6. ML Model 7. SMOTE Function 	



IBM corporate employee attrition analysis.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Code + Test

```

from prettytable import prettytable

Function for HyperParameter Tuning

[] def bestHyperFunc(algo, x_train, y_train, x_cv, y_cv, x_test, y_test, verbose, kernel_select=None):

    print(kernel_select)

    #defining values for hyperparameter
    alpha = [10**x for x in range(-4, 1)]
    C_log=[math.log(i) for i in alpha]
    #returning error array
    cv_log_error_array = []

    for i in alpha:

        if(verbose==1):
            print('for alpha =', i)

        if(algo=='logistic_regression'):
            clf = SGDClassifier(alpha, penalty='l1', loss='log', random_state=42)

        elif(algo=='svm'):
            clf = SVC(C=C(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.ordb = sig_clf.predict_proba(x_test)

```

```
bm corporate employee attrition analysis.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Test
Connect + / Editing ^

... (page=0 or 1)
clf = SK(cv,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_proba = sig_clf.predict_proba(x_cv)
cv_log_error_array.append(log_loss(x_cv, sig_clf_proba, labels=clf.classes_, eps=1e-15))
if(verbose==1):
    print("log loss : ", log_loss(x_cv, sig_clf_proba))

if(verbose==1):
    fig, ax = plt.subplots()
    ax.plot(cv_log_cv_log_error_array,cv_gamma)
    for i, txt in enumerate(np.round(cv_log_error_array,3)):
        ax.annotate((i,txt), (i,txt), (i,log(1,np.round(cv_log_error_array[1,2]))))
    plt.grid()
    plt.title("Cross validation error for each alpha")
    plt.xlabel("alpha (gamma)")
    plt.ylabel("error measure")
    plt.show()

return cv_log_error_array,alpha

+ designing ML Algo For the Best Fit HyperParameter Value

[ ] def main_model(alpha,x_train,y_train,x_cv,y_cv,x_test,y_test,verbose,kernel_select=None):
    """
    Evaluating besthyperparameter and that function return's the CVlog Error Array, alpha
    cv_log_error_array,alpha = besthyperparameter(alpha,x_train,y_train,x_cv,y_cv,x_test,y_test,verbose,kernel_select)

    """
```

```
bm corporate employee attrition analysis.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Test
Connect + / Editing ^

return cv_log_error_array,alpha

+ designing ML Algo For the Best Fit HyperParameter Value

[ ] def main_model(alpha,x_train,y_train,x_cv,y_cv,x_test,y_test,verbose,kernel_select=None):
    """
    Evaluating besthyperparameter and that function return's the CVlog Error Array, alpha
    cv_log_error_array,alpha = besthyperparameter(alpha,x_train,y_train,x_cv,y_cv,x_test,y_test,verbose,kernel_select)

    """
    print(kernel_select)

    print(cv_log_error_array,alpha)
    print("*****besthyperparameter Completed*****")

    #finding best HyperParameter Index from cv_log_error_array using np.argmin()
    #now that index find its value
    best_alpha = np.argmin(cv_log_error_array)
    print("for best_alpha = ", alpha[best_alpha])

    #actual ML algorithm naming
    if(verbose==1):
        clf = SK(cv_log_error_array[best_alpha],kernel_kernel_select,probability=True, class_weight='balanced')

    else(verbose==0):
        clf = SK(cv_log_error_array[best_alpha],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)
```

```
bm corporate employee attrition analysis.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Test
Connect + / Editing ^

... (page=0 or 1)
clf = SK(cv_log_error_array[best_alpha],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)

#predicting output labels for all train,cv and Test Data
predict_train=sig_clf.predict(y_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:",logloss_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 loss 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:",logloss_test)

if(verbose==1):
    #calling plotting function
    plot_confusion_matrix(y_test,predict_test)

return predict_train,predict_cv,predict_test,logloss_train,logloss_cv,logloss_test,clf
```