
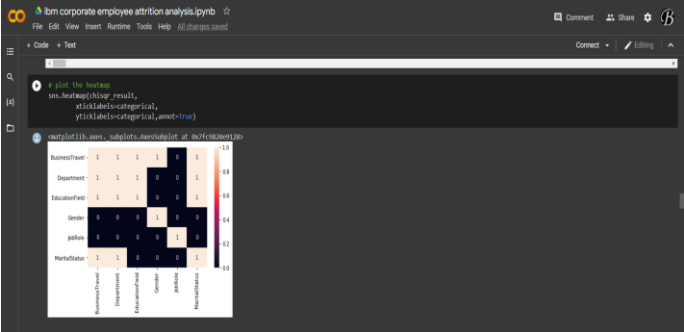
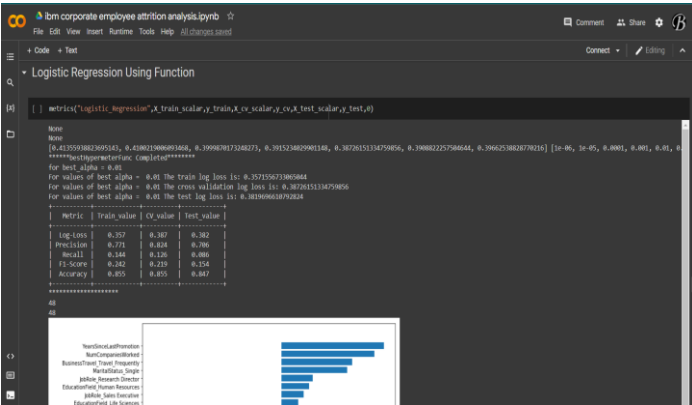


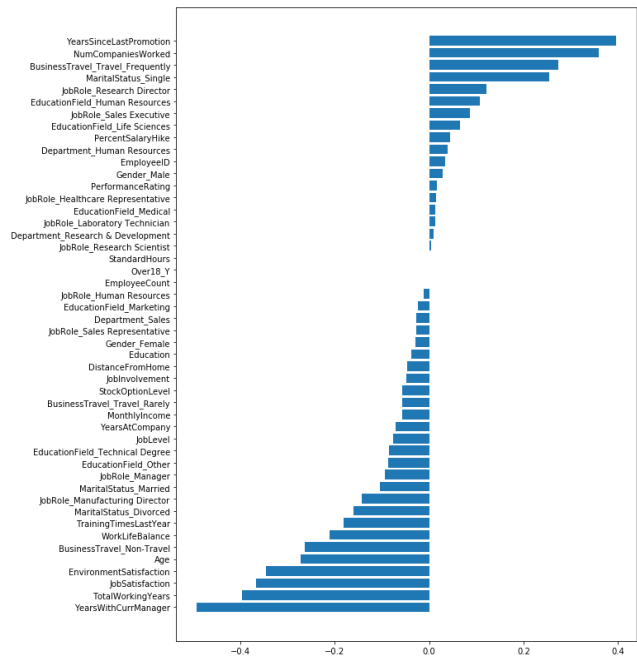
Project Development Phase Model Performance Test

Date	10 November 2022
Team ID	PNT2022TMID37447
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
1.	Metrics	<p>Regression Model: R2 score -79%</p> <p>Classification Model: Confusion Matrix - 79% Accuracy Score- 76%</p> <p>& Classification Report –</p> <ol style="list-style-type: none"> 1. Correlation 2. Confusion matrix 3. Logistic regression 4. Linear SVM 5. RBF Kernel 6. Poly Kernel 	  



```

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

+ Code + Test
Connect + Editing

SVM
Using Linear Kernel

[ ] main_model("SVM",X_train_scaled,y_train,X_cv_scaled,y_cv,X_test_scaled,y_test,"linear")
metrics("SVM",X_train_scaled,y_train,X_cv_scaled,y_cv,X_test_scaled,y_test,"linear")

linear
linear
[0.400126025026025, 0.400126025026025, 0.400126025026025, 0.39844055688551, 0.39233542092079, 0.3943253431171, 0.392734725343513] [1e-06, 1e-05, 0.0001, 0.001, 0.01, 0.1, 1]
*****testHyperparameter: Complete*****
for best alpha = 0.01
for values of best alpha = 0.01 the train log loss is: 0.3065780274237655
for values of best alpha = 0.01 the cross validation log loss is: 0.39233542092079
for values of best alpha = 0.01 the test log loss is: 0.3933836222945994

Metric | train_value | cv_value | test_value |
-----|-----|-----|-----|
log_loss | 0.361 | 0.392 | 0.38 |
Precision | 0.652 | 0.75 | 0.5 |
Recall | 0.497 | 0.135 | 0.409 |
F1-Score | 0.348 | 0.229 | 0.157 |
Accuracy | 0.646 | 0.653 | 0.618 |

*****
NonClassifiedPredictions
NumCompaniesWorked

```

```

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

+ Code + Test
Connect + Editing

Using RBF Kernel

[ ] main_model("SVM",X_train_scaled,y_train,X_cv_scaled,y_cv,X_test_scaled,y_test,"rbf")
metrics("SVM",X_train_scaled,y_train,X_cv_scaled,y_cv,X_test_scaled,y_test,"rbf")

rbf
rbf
[0.388347918218025, 0.388347918218025, 0.388347918218025, 0.388347918218025, 0.388347918218025, 0.388347918218025, 0.388347918218025] [1e-06, 1e-05, 0.0001, 0.001, 0.01, 0.1, 1]
*****testHyperparameter: Complete*****
for best alpha = 1
for values of best alpha = 1 the train log loss is: 0.345008077516388
for values of best alpha = 1 the cross validation log loss is: 0.3954478043576381
for values of best alpha = 1 the test log loss is: 0.395977638156432

Metric | train_value | cv_value | test_value |
-----|-----|-----|-----|
log_loss | 0.348 | 0.255 | 0.22 |
Precision | 0.493 | 0.77 | 0.439 |
Recall | 0.915 | 0.484 | 0.676 |
F1-Score | 0.583 | 0.637 | 0.749 |
Accuracy | 0.568 | 0.567 | 0.537 |

*****
Starting Get Important Features Function
rbf
kernel is rbf so, we cannot get the important features using coef_ function

```

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

+ Code + Test
kernel is rbf so, we cannot get the important features using Coef_ function

Using Poly Kernel

def hyperparameterfunc("SVM", X_train, y_train, X_cv, y_cv, X_test, y_test, poly):
    main_model = svm.SVC(kernel=poly, C=1, gamma=0.001)
    metrics["SVM", X_train, y_train, X_cv, y_cv, X_test, y_test, poly]

poly
[0.4268913254648181, 0.4268913252368895, 0.4268913251488913, 0.4268913256818227, 0.42651262675687884, 0.3172793674744845, 0.21127574286677665] [1e-05, 1e-05, 0.0001, 0.001, 0.01, 0.1]
****HyperparameterFunc Completed****
For best alpha = 1
For values of best alpha = 1 The train log loss is: 0.111262149113629316
For values of best alpha = 1 The cross validation log loss is: 0.21127574286677665
For values of best alpha = 1 The test log loss is: 0.21036423825263828
-----
| Metric | Train value | Cv value | Test value |
-----
| log-loss | 0.111 | 0.211 | 0.193 |
| Precision | 0.954 | 0.88 | 0.919 |
| Recall | 0.938 | 0.293 | 0.764 |
| F1-Score | 0.941 | 0.434 | 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_ function
```

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

+ Code + Test
kernel is rbf so, we cannot get the important features using Coef_ function

With Balanced Data : For Balancing Data using SMOTE function from Imblearn Package

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)))

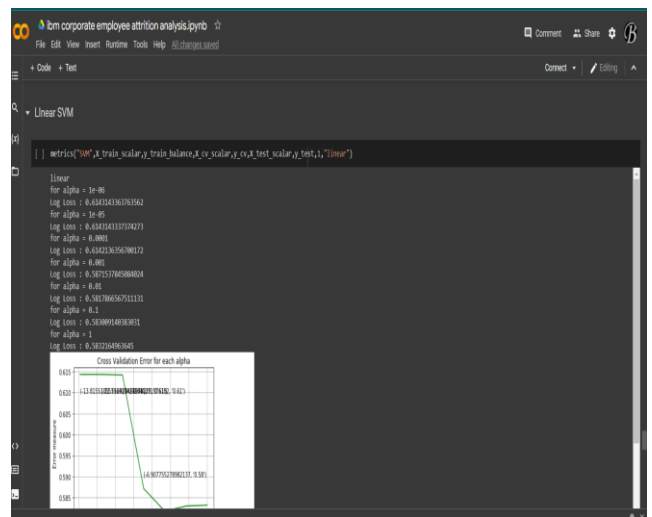
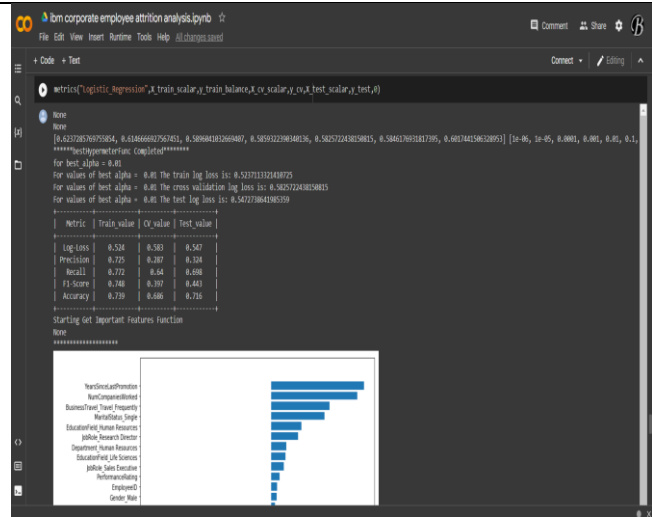
sm = SMOTE(random_state=0)
X_train_balance, y_train_balance = sm.fit_sample(X_train, y_train.ravel())

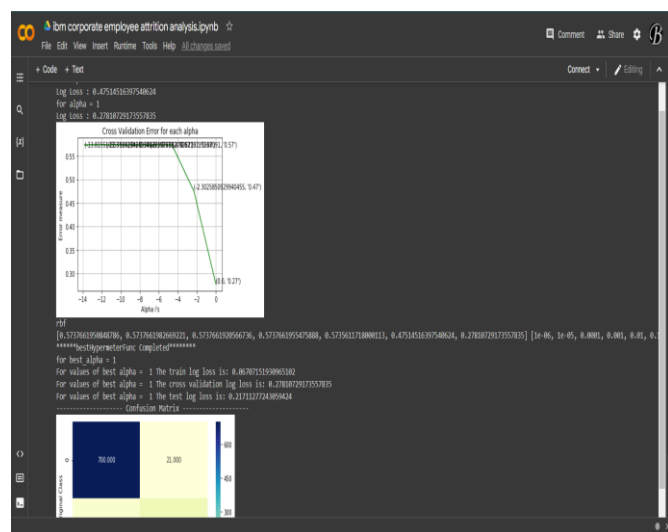
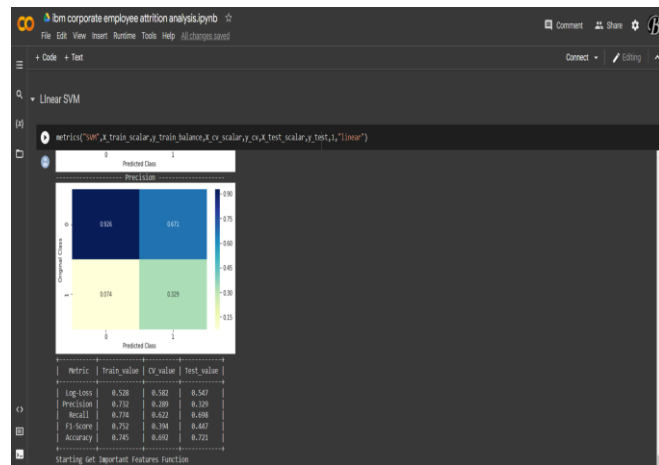
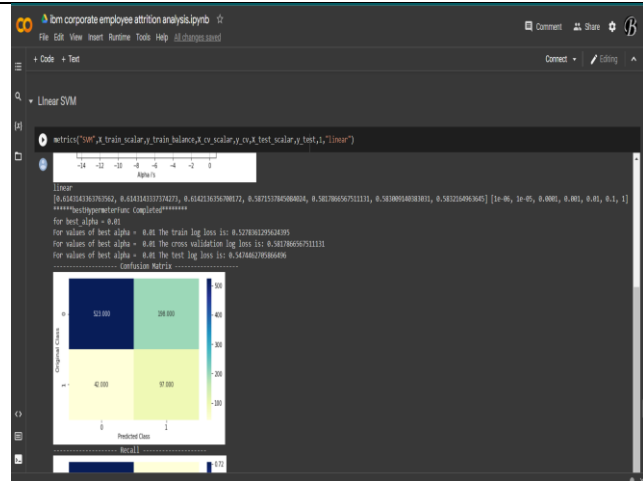
print("After oversampling, the shape of train X: {}".format(X_train_balance.shape))
print("After oversampling, the shape of train y: {}".format(y_train_balance.shape))

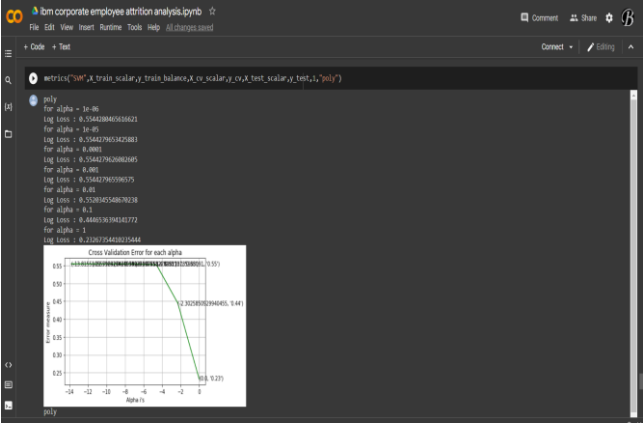
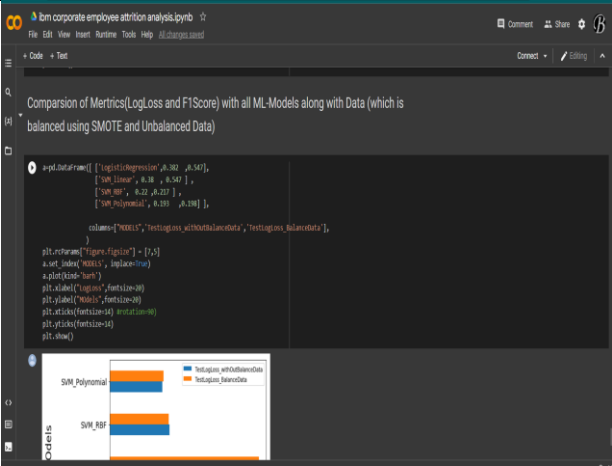
print("After oversampling, counts of Label '1': {}".format(sum(y_train_balance==1)))
print("After oversampling, counts of Label '0': {}".format(sum(y_train_balance==0)))

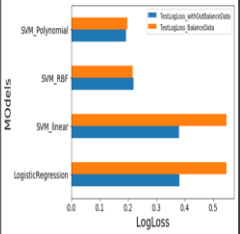
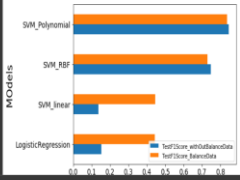
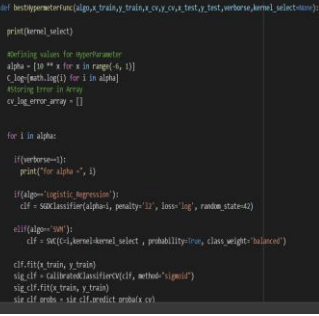
Before oversampling, counts of Label '1': 485
Before oversampling, counts of Label '0': 2387
After oversampling, the shape of train X: (4814, 48)
After oversampling, the shape of train y: (4814,)
After oversampling, counts of Label '1': 2387
After oversampling, counts of Label '0': 2387

[ ] scaler = StandardScaler()
scaler.fit(X_train_balance)
X_train_scaled=scaler.transform(X_train_balance)
X_cv_scaled=scaler.transform(X_cv)
X_test_scaled=scaler.transform(X_test)
```





			
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 	

			 <pre> a.set_index('models', inplace=True) a.plot(kind='bar') plt.xlabel("LogLoss", fontsize=30) plt.ylabel("Models", fontsize=30) plt.xticks(fontsize=14, rotation=90) plt.yticks(fontsize=14) plt.show() </pre> <pre> [] a.plot_dataframe([['LogisticRegression', 0.154, 0.443], ['SVM_Linear', 0.137, 0.447], ['SVM_RBF', 0.109, 0.37], ['SVM_Polynomial', 0.105, 0.337]], columns=['Model', 'testScore_withOutBalanceData', 'testScore_BalanceData'], plot_kwarg={'figure(figsize)=(7,5)}) a.set_index('models', inplace=True) a.plot(kind='bar') plt.xlabel("LogLoss", fontsize=30) plt.ylabel("Models", fontsize=30) plt.xticks(fontsize=14, rotation=90) plt.yticks(fontsize=14) plt.show() </pre>
			 <pre> a.set_index('models', inplace=True) a.plot(kind='bar') plt.xlabel("LogLoss", fontsize=30) plt.ylabel("Models", fontsize=30) plt.xticks(fontsize=14, rotation=90) plt.yticks(fontsize=14) plt.show() </pre>
			 <pre> def bestHyperparameter(alpha_train, y_train, x_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 = [math.log(i) for i in alpha] #defining error array cv_log_error_array = [] for i in alpha: if(verbose==0): 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=False, class_weight='balanced') clf.fit(x_train, y_train) sig_clf = CalibratedClassifer(clf, method='sigmoid') sig_clf.fit(x_train, y_train) sig_clf_preds = sig_clf.predict_proba(x_cv) </pre>

			The notebook also shows a section titled "designing ML Algo For the Best Fit HyperParameter Value" with a function definition: <pre> def main_model(alpha, x_train, y_train, x_cv, y_cv, x_test, y_test, verbose, kernel, select=None): #calling besthyperparameter and that function return's the (cv_log_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) #printing kernel select print(kernel_select) #printing cv_log_error_array, alpha print(cv_log_error_array, alpha) print("*****besthyperparameter (completed)*****") #finding best hyperparameter index from cv_log_error_array using np.argmin() #since 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(alpha=="logistic regression"): clf = LogisticRegression(penalty='l2', loss='log', random_state=42) else(alpha=="svm"): clf = SVC(kernel=kernel, probability=True, class_weight='balanced') clf.fit(x_train, y_train) sig_clf = CalibratedClassifierCV(clf, method='sigmoid') sig_clf.fit(x_train, y_train) </pre>