Team Presentation II: Predictive Model

MSBA Cohort 1: {Monday/Wednesday}

Team 11

Team Members:

1) Handi Wang
2) Tongxin Guo
3) Kunyi Shi
4) Qi Yue
5) Pramath Kashyap

Project Outline

In this assignment, we have to develop a predictive model and a decision support system (DSS) that evaluates the risk of Home Equity Line of Credit (HELOC) applications.

Data

The dataset and additional information can be found on https://community.fico.com/s/explainable-machinelearning-challenge. This also excel sheets along with its decision rules.

Details

Using the dataset, we developed a predictive model to assess credit risk. We compared various models and different algorithms and shortlisted the most appropriate algorithm for prediction depending upon the accuracy. We have also created a prototype of an interactive interface that sales representatives in a bank/credit card company can use to decide on accepting or rejecting applications.

Approach:

Step 1: Importing Libraries

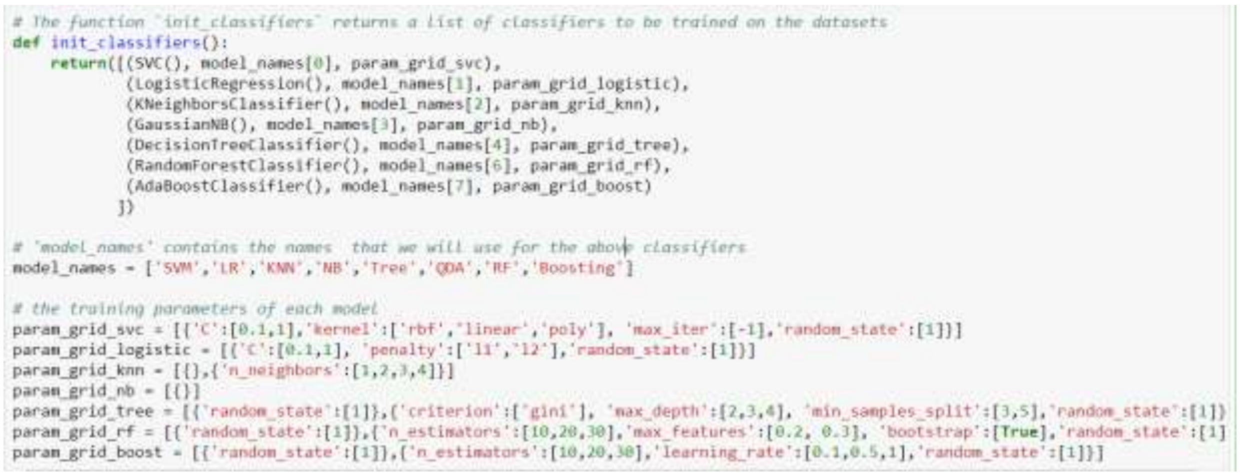
Step 2: Data Cleaning Data Dictionary

Firstly, we see the percentage of pf -7, -8, -9 of the total rows. We drop the rows containing special values such as -9. As we have a significant number of values of -7 & -8, dropping those values would result in less number of datapoints which would result in in accurate evaluation of our models. Besides, if we take average of these special values, it would result in complete change of original meaning of these values. Hence, we could not move forward with same. Thus, post initial research we can say that the special values -7, -8 does have certain relevance in order to arrive at our results. Further, we separated Independent (X) variables from Independent Variables (Y). We considered all the variables except RiskPerformance as an independent variables and Risk Performance was the predicted variable.

Post separation, we had reconstructed the variables Max Delqz Public'/ 'Max Delq Ever' using get_dummies and we were able to recreate Independent Variables. We do so because the category represents delinquency status and not all values are numeric. They are majorly categorical variables.

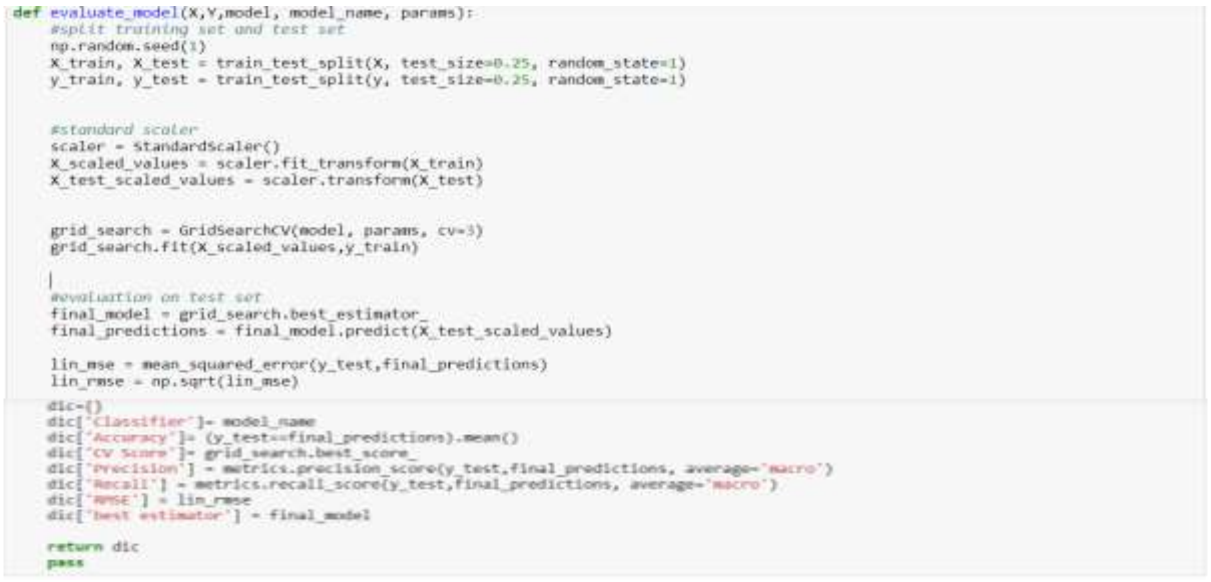
Steps 3: Modeling and Evaluation

We used various models to create prediction and followed by certain parameters such RSME and accuracy to evaluate these models. Some of the models included SVM, Logistic Regression KNN, Gaussian, Decision Tree classifier, Random Forest, Ada Boost classifier. We had evaluated these models on the lines of methods we learned during our classes. We had splitted the data set into training and testing data on a loose ratio of 75:25 split.



```python
# The function `init_classifiers` returns a list of classifiers to be trained on the datasets
def init_classifiers():
    return([(SVC(), model_names[0], param_grid_svc),
            (LogisticRegression(), model_names[1], param_grid_logistic),
            (KNeighborsClassifier(), model_names[2], param_grid_knn),
            (GaussianNB(), model_names[3], param_grid_nb),
            (DecisionTreeClassifier(), model_names[4], param_grid_tree),
            (RandomForestClassifier(), model_names[6], param_grid_rf),
            (AdaBoostClassifier(), model_names[7], param_grid_boost)
           ])

# 'model_names' contains the names  that we will use for the above classifiers
model_names = ['SVM','LR','KNN','NB','Tree','QDA','RF','Boosting']

# the training parameters of each model
param_grid_svc = [{'C':[0.1,1],'kernel':['rbf','linear','poly'], 'max_iter':[-1],'random_state':[1]}]
param_grid_logistic = [{'C':[0.1,1], 'penalty':['l1','l2'],'random_state':[1]}]
param_grid_knn = [{},{'n_neighbors':[1,2,3,4]}]
param_grid_nb = [{}]
param_grid_tree = [{'random_state':[1]},{'criterion':['gini'], 'max_depth':[2,3,4], 'min_samples_split':[3,5],'random_state':[1]}
param_grid_rf = [{'random_state':[1]},{'n_estimators':[10,20,30],'max_features':[0.2, 0.3], 'bootstrap':[True],'random_state':[1]
param_grid_boost = [{'random_state':[1]},{'n_estimators':[10,20,30],'learning_rate':[0.1,0.5,1],'random_state':[1]}]
```

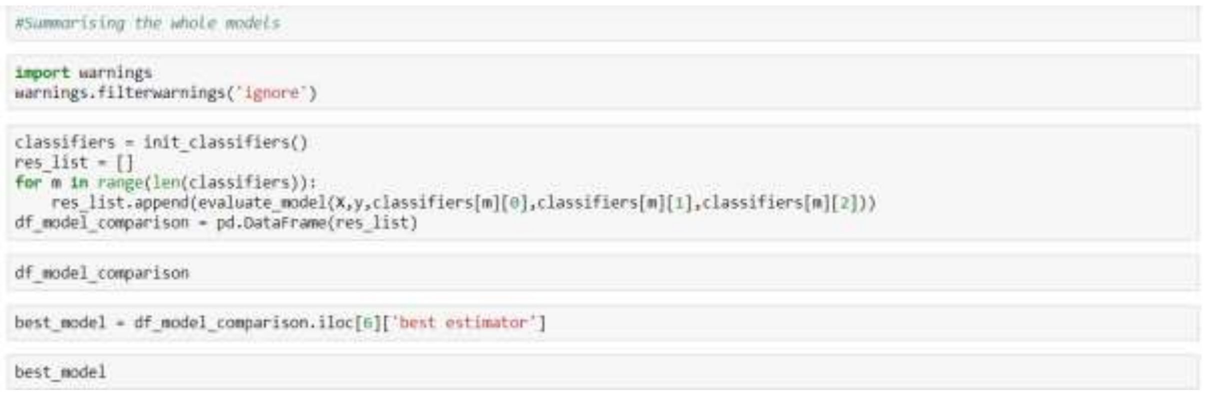
We used standard scalar to scale the independent variables.



```python
def evaluate_model(X,Y,model, model_name, params):
    #split training set and test set
    np.random.seed(1)
    X_train, X_test = train_test_split(X, test_size=0.25, random_state=1)
    y_train, y_test = train_test_split(y, test_size=0.25, random_state=1)


    #standard scaler
    scaler = StandardScaler()
    X_scaled_values = scaler.fit_transform(X_train)
    X_test_scaled_values = scaler.transform(X_test)


    grid_search = GridSearchCV(model, params, cv=3)
    grid_search.fit(X_scaled_values,y_train)

    
    #evaluation on test set
    final_model = grid_search.best_estimator_
    final_predictions = final_model.predict(X_test_scaled_values)

    lin_mse = mean_squared_error(y_test,final_predictions)
    lin_rmse = np.sqrt(lin_mse)
    dic={}
    dic['Classifier']= model_name
    dic['Accuracy']= (y_test==final_predictions).mean()
    dic['CV Score']= grid_search.best_score_
    dic['Precision'] = metrics.precision_score(y_test,final_predictions, average='macro')
    dic['Recall'] = metrics.recall_score(y_test,final_predictions, average='macro')
    dic['RMSE'] = lin_rmse
    dic['best estimator'] = final_model

    return dic
    pass
```

Moving forward, we used Hyper parameter tuning using GridSearch CV in order to find the best model among all. We created data frame which containing each of the model. Post this, we calculated the values of Root Mean Square error, accuracy, recall, precision and CV score to choose the best model having best accuracy and least error.

Lastly, we summarized the whole models and checked the value of best model.



```python
#Summarising the whole models
```

```python
import warnings
warnings.filterwarnings('ignore')
```

```python
classifiers = init_classifiers()
res_list = []
for m in range(len(classifiers)):
    res_list.append(evaluate_model(X,y,classifiers[m][0],classifiers[m][1],classifiers[m][2]))
df_model_comparison = pd.DataFrame(res_list)
```

```python
df_model_comparison
```

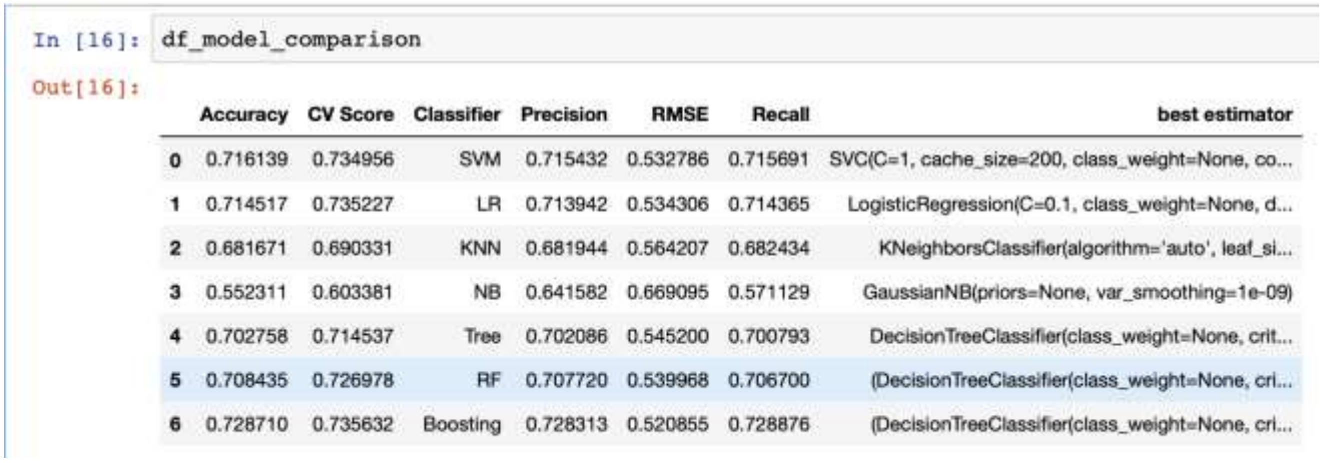
```python
best_model = df_model_comparison.iloc[6]['best estimator']
```

```python
best_model
```

We saw Adaboost to be best model among all as the accuracy was the highest and error was the least among all.

Why did we select the AdaBoost model?

As we see from below, the accuracy of Ada Boost is highest which is 0.7287 and that the RMSE is the the lowest which is 0.520855 among other models. We also took into consideration Precision and Recall for arriving at better results.
<u>Precision</u>: The percentage of retrieved true positive intances among all predicted positive instances:
Recall (True Positive Rate): The percentage of retrieved true positive instances among all true positive instances.

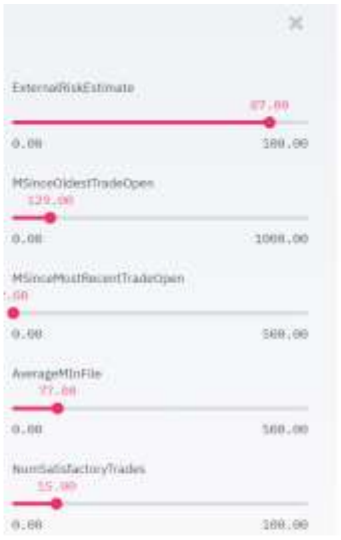


```
In [16]: df_model_comparison
Out[16]:
```

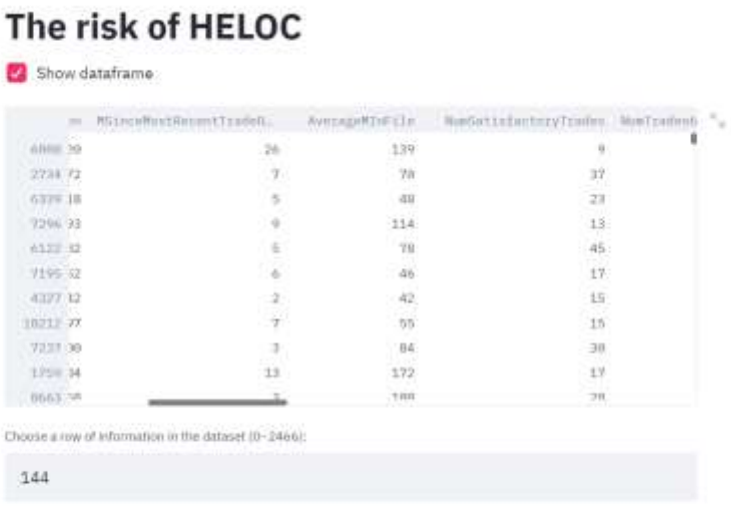
	Accuracy	CV Score	Classifier	Precision	RMSE	Recall	best estimator
0	0.716139	0.734956	SVM	0.715432	0.532786	0.715691	SVC(C=1, cache_size=200, class_weight=None, co...
1	0.714517	0.735227	LR	0.713942	0.534306	0.714365	LogisticRegression(C=0.1, class_weight=None, d...
2	0.681671	0.690331	KNN	0.681944	0.564207	0.682434	KNeighborsClassifier(algorithm='auto', leaf_si...
3	0.552311	0.603381	NB	0.641582	0.669095	0.571129	GaussianNB(priors=None, var_smoothing=1e-09)
4	0.702758	0.714537	Tree	0.702086	0.545200	0.700793	DecisionTreeClassifier(class_weight=None, crit...
5	0.708435	0.726978	RF	0.707720	0.539968	0.706700	(DecisionTreeClassifier(class_weight=None, cri...
6	0.728710	0.735632	Boosting	0.728313	0.520855	0.728876	(DecisionTreeClassifier(class_weight=None, cri...

<u>Interface</u>

An interface was developed using in order to represent respective models. We did data cleaning using pipeline and stream lit package for the interface.
We can input the row of information in the dataset using the side bars or the user can select the one of the rows in the data frames to make the prediction.



ExternalRiskEstimate 87.00 (0.00 – 100.00)
MSinceOldestTradeOpen 129.00 (0.00 – 1000.00)
MSinceMostRecentTradeOpen .00 (0.00 – 500.00)
AverageMInFile 77.00 (0.00 – 500.00)
NumSatisfactoryTrades 15.00 (0.00 – 100.00)



The risk of HELOC

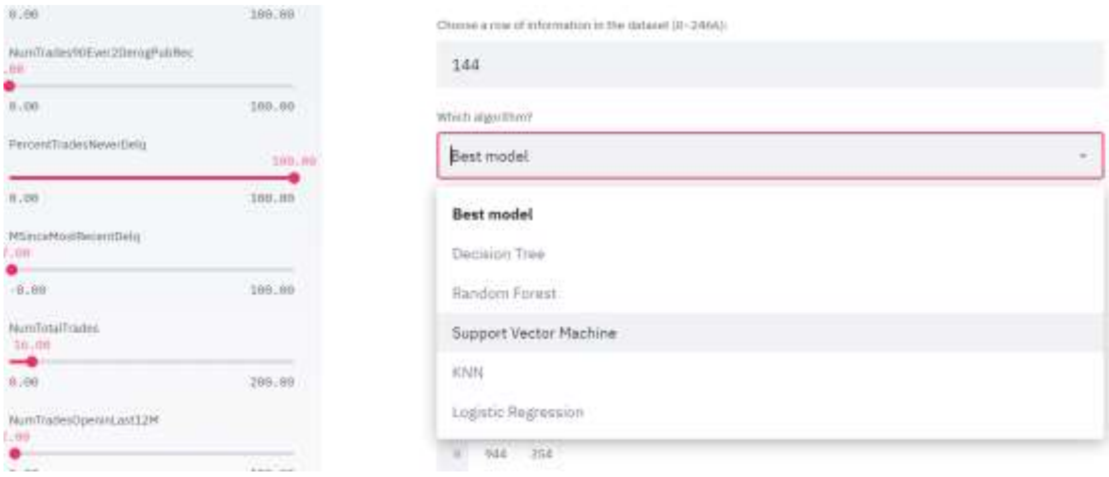
[✓] Show dataframe

	...	MSinceMostRecentTradeO...	AverageMInFile	NumSatisfactoryTrades	NumTrades...
6888	39	26	139	9	
2734	72	7	78	37	
6329	18	5	48	23	
7296	33	9	114	13	
6122	32	5	78	45	
7195	52	6	46	17	
4327	12	2	42	15	
10212	77	7	55	15	
7237	30	3	84	38	
1758	34	13	172	17	
8663	38	3	188	28	

Choose a row of information in the dataset (0-2466):

144

User can select the algorithms based on their preference and comparing it with other models


NumTrades90Ever2DerogPubRec: 0.00 (0.00 – 100.00)

PercentTradesNeverDelq: 100.00 (0.00 – 100.00)

MSinceMostRecentDelq: 7.00 (-8.00 – 100.00)

NumTotalTrades: 16.00 (0.00 – 200.00)

NumTradesOpeninLast12M: 1.00

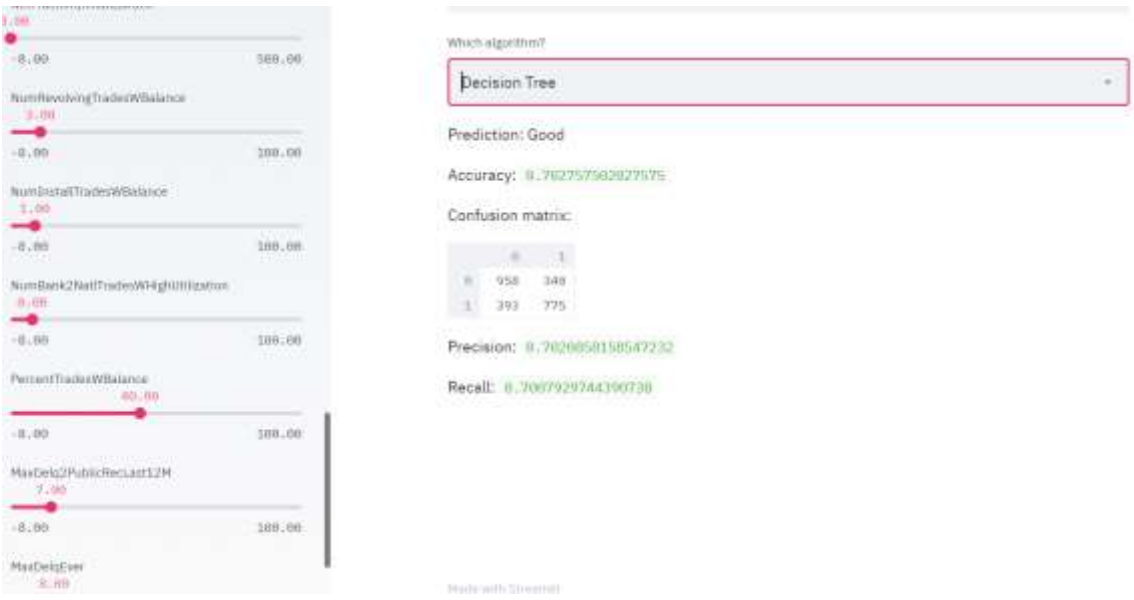
Choose a row of information in the dataset (0-2464):

144

Which algorithm?

Best model

We receive output containing prediction, accuracy, confusion matrix, precision and recall


NumRevolvingTradesWBalance: 3.00 (-8.00 – 100.00)

NumInstallTradesWBalance: 1.00 (-8.00 – 100.00)

NumBank2NatlTradesWHighUtilization: 0.00 (-8.00 – 100.00)

PercentTradesWBalance: 40.00 (-8.00 – 100.00)

MaxDelq2PublicRecLast12M: 7.00 (-8.00 – 100.00)

MaxDelqEver: 8.00

Which algorithm?

Decision Tree

Prediction: Good

Accuracy: 0.702757582827575

Confusion matrix:

	0	1
0	958	340
1	393	775

Precision: 0.7020858158547232

Recall: 0.7007929744390738

Made with Streamlit

<u>Conclusion</u>

```
best_model
```

```
AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None, learning_rate=1.0,
                   n_estimators=50, random_state=1)
```

Thus, we can say that AdaBoost Classifier is best prediction model that evaluates the risk of Home Equity Line of Credit (HELOC) applications