Capstone Project - 2

Team Space: Company Bankruptcy Prediction

Team Members:

Saubhagya Verma

Harsh Mudgil

Tawheed Yousuf

Harshal Pawar

Jimmi Kumar

Sai Krishna Reddy Palle

Catching the Doom, Before it Happens

1. Problem Statement
2. Feature Selection
3. Exploratory Data Analysis
4. Applying the Models
5. Model Selection and validation

Problem Statement

- Prediction of bankruptcy is a phenomenon of increasing interest to firms who stand to lose money because of unpaid debts. Since computers can store huge dataset pertaining to bankruptcy, making accurate predictions from them beforehand is becoming important.

- A highly unbalanced dataset, to predict the financial state of the companies in Taiwan, was presented. Company bankruptcy was defined based on the business regulations of the Taiwan Stock Exchange.

- For this project, we have aimed to curate a model that captures the bankruptcy patterns among companies in the industry. This model will work to provide early signs of financial downturn in the corporations

➢ Data Pipeline

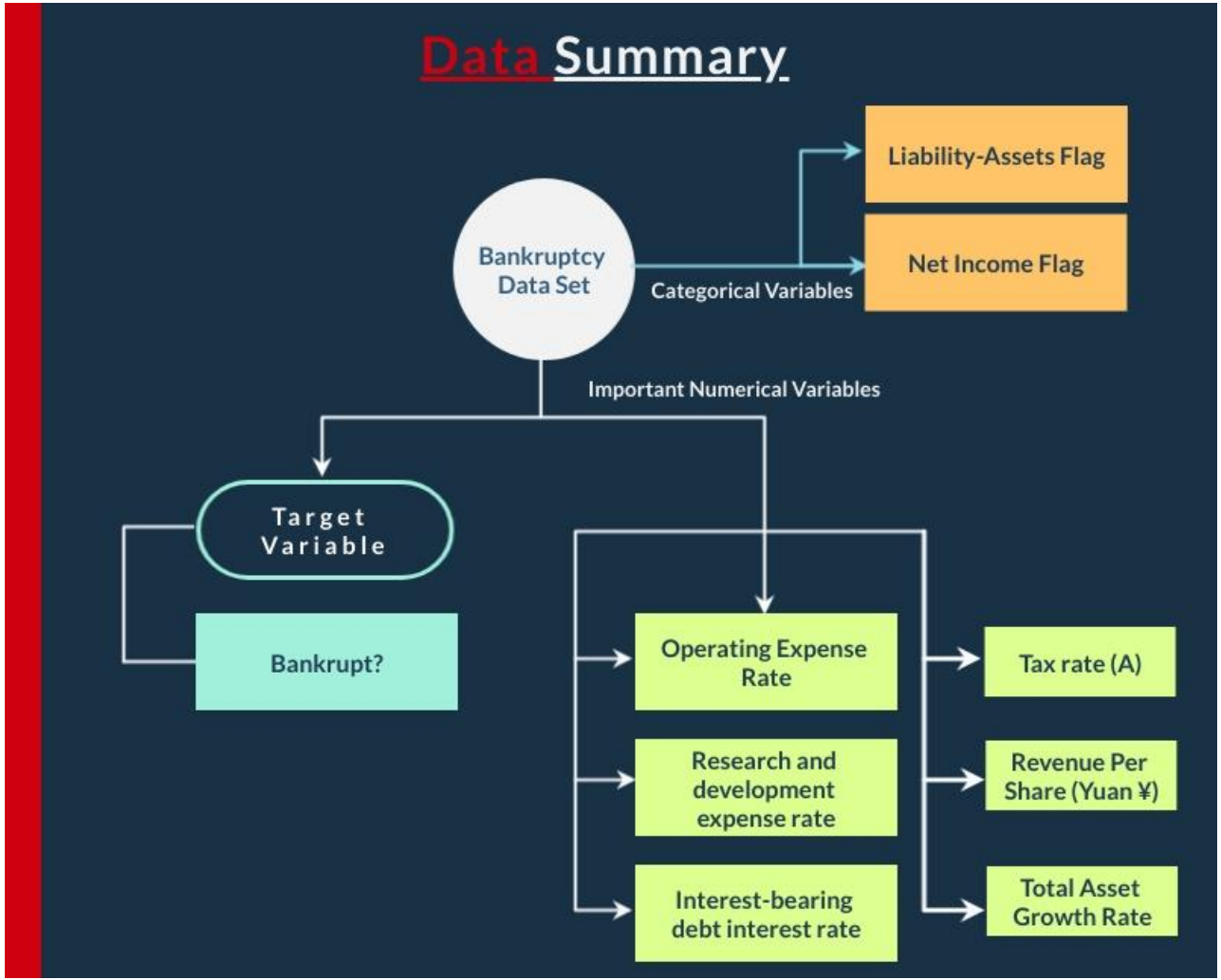
- **Cleaning the Data:** The data was checked for null values, categorical values and primary inspection was performed.
- **Feature Selection:** Techniques such as VIF, p-value, L-1 Regularization and Information Gain were performed to select important features.
- **EDA:** Exploratory analysis was performed to review the skewness in the data, outliers and correlation patterns.
- **Model Testing:** Combination of different models and feature selection techniques were used to determine optimal results

Data Pipeline

- The data were collected from the Taiwan Economic Journal for the years 1999 to 2009. Company bankruptcy was defined based on the business regulations of the Taiwan Stock Exchange.

- The dataset consisted of 96 columns with mainly of continuous features in 6819 rows

Data Summary



➢ Data Summary

- **Operating Expense Rate:** The operating expense rate shows the efficiency of a company's management by comparing the total operating expense (OPEX) of a company to net sales.

- **Research and Development Expense Rate:** Research and development (R&D) expenses are associated directly with the research and development of a company's goods or services and any intellectual property generated in the process.

- **Interest Bearing Debt Interest Rate:** The interest-bearing debt ratio is significant because it gives a window into the financial health of a company. The interest-bearing debt ratio, or debt to equity ratio, is calculated by dividing the total long-term, interest-bearing debt of the company by the equity value.

Data Summary

- **Tax Rate:** A tax rate is the percentage at which an individual or corporation is taxed.

- **Revenue per share:** Amount of revenue over common shares outstanding. Answers the question, what's the ownership of sales to each share? Increasing revenue per share (RPS) over time is a good sign, because it means each share now has claim to more revenues.

- **Total Asset Growth Rate:** Total Asset Growth Rate defined as year-over-year percentage change in total assets

Important Feature Selection Techniques

➢ Important Feature Selection Techn.

- **Quasi Method:** Quasi-constant features are those that show the same value for the great majority of the observations of the dataset. In general, these features provide little if any information that allows a machine learning model to discriminate or predict a target.

  Using Quasi Constant Method: 31 columns were selected

- **Lasso Regression:** In linear model regularization, the penalty is applied over the coefficients that multiply each of the predictors. Lasso or l1 has the property that is able to shrink some coefficients to zero. Therefore, that feature can be removed from the model.

Using Lasso Regression Method and VIF scores: 19 columns were selected

➢ Important Feature Selection Techn.

- **Information Gain:** Information gain or mutual information measures how much information the presence/absence of a feature contributes to making the correct prediction on the target.

  Using Information Gain Method: 30columns were selected

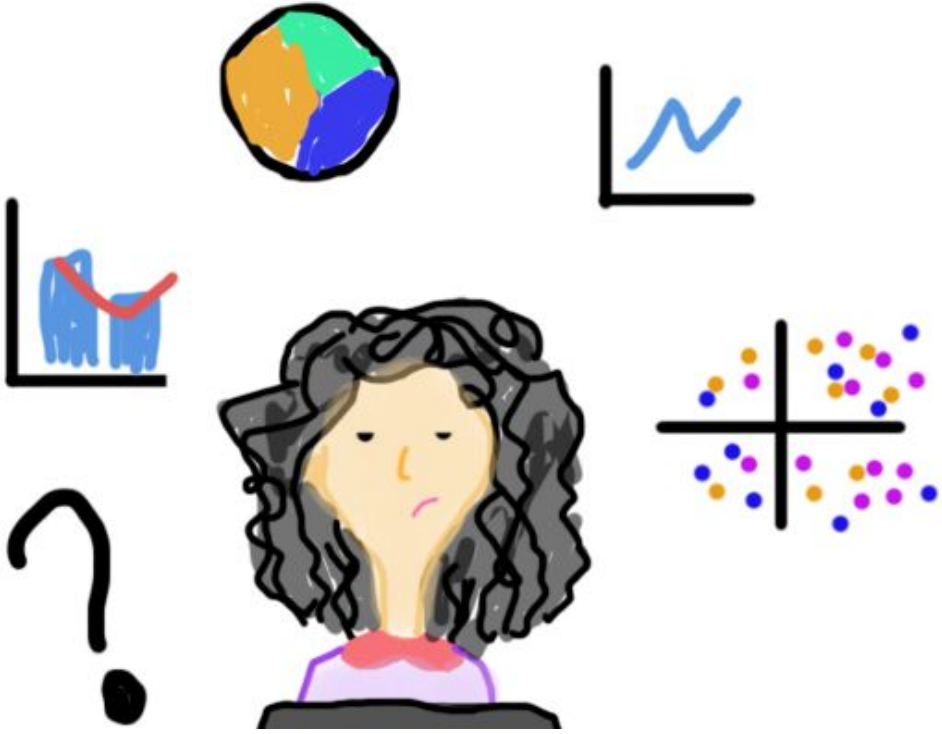
- **Random Forest:** Random forests consist of 4-12 hundred decision trees, each of them built over a random extraction of the observations from the dataset and a random extraction of the features. Features that are selected at the top of the trees are in general more important than features that are selected at the end nodes of the trees, as generally the top splits lead to bigger information gains.

  Using Random Forest: 30 columns were selected

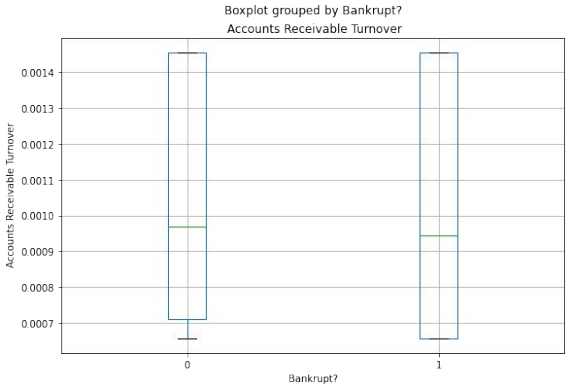
➢ Other Feature Selection Techn.

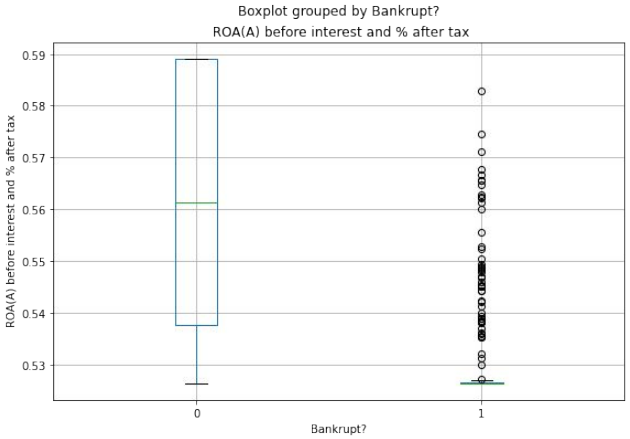
1. Using VIF- 71 columns were selected.
2. Using VIF and p_values(logit) 14 were selected.
3. Using p_values- OLS 32 were selected.
4. Using just Lasso -24.

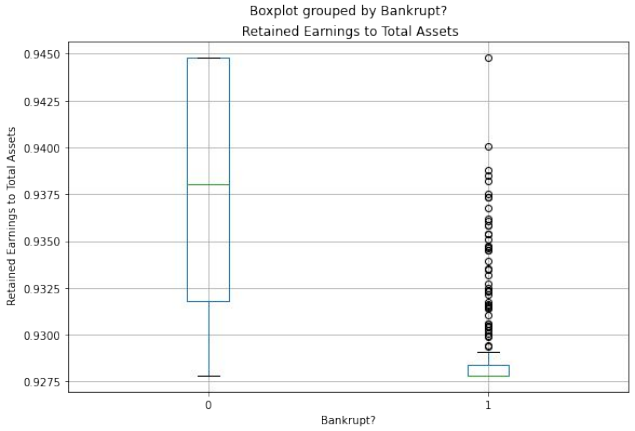
Exploratory Data Analysis



Primary EDA: Understanding the effect of the value of a feature on the decision



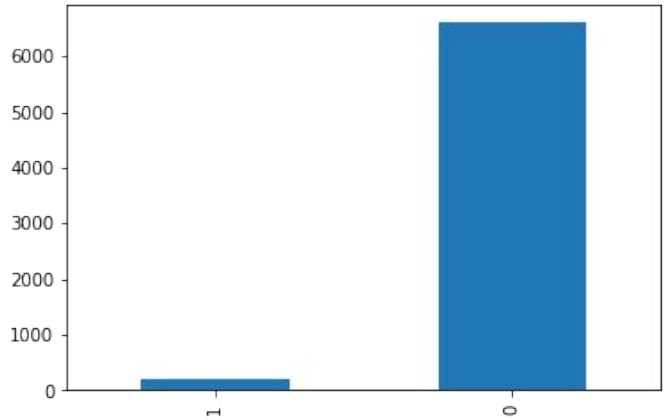




➢ Insights

- Bankruptcy is more likely, if the value of features such as ROA(A) value is low.
- Value of features such as Accounts Receivable Turnover is less likely to cause bankruptcy.
- For higher values of features like Retained Earnings to Total Assets, a company is likely to stay afloat.

Primary EDA: Checking the Imbalance




```
df['Bankrupt?'].value_counts()
```

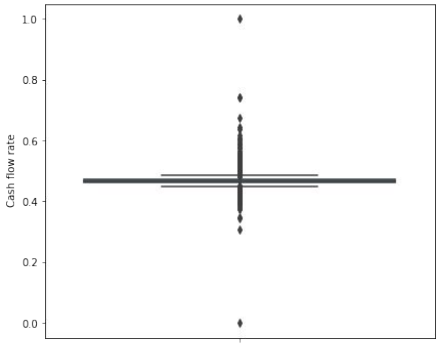
```
0    6599
1     220
Name: Bankrupt?, dtype: int64
```

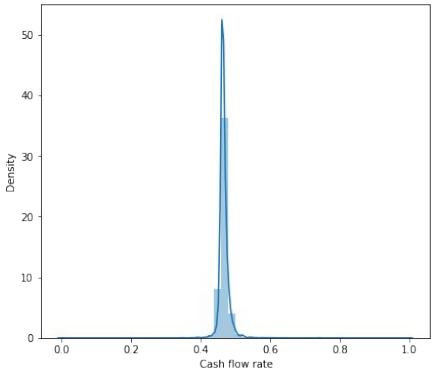
➢ Insights

- **The target variable is highly imbalanced. Less than 3.5% instances for bankrupt companies exist in the data set.**

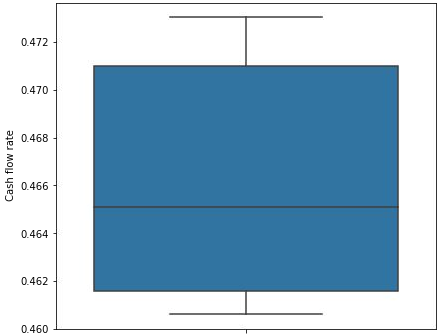
Primary EDA: Detecting and Capping outliers

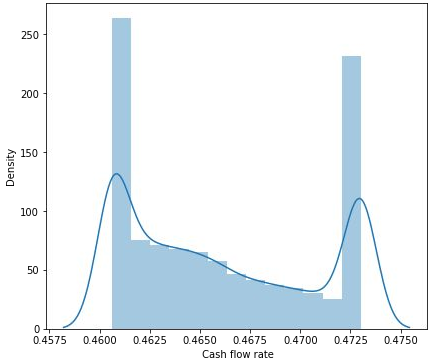
Before





After





- **Outliers were capped at 80 percentile value on the upper side and at 20 percentile value at the bottom side**

Primary EDA: Understanding the correlation between features and target variable

Primary EDA: Understanding the correlation between features and target variable. (Contd.)

➢ Insights

- After going through all the columns, we found that none of the features displayed any high correlation(> 0.5) with the target variable, bankruptcy.
- The most negatively correlated feature was ' Net-Income to Total Assets' i.e -0.32
- The most positively correlated feature was ' Debt Ratio' i.e +0.25.
- Conclusion: Our dataset doesn't provide any modelling power with respect to linear algorithms. Since there is a high imbalance in the classes, this looks like an anomaly detection problem. So, let us try anomaly detection algorithm.

EDA: Using Anomaly Detection Algorithm to further explore the data.

➢ Insights

- Isolation Forest was used to detect anomalies in the dataset. Since results, were not up to the mark, let us visualize the dataset to investigate the problem

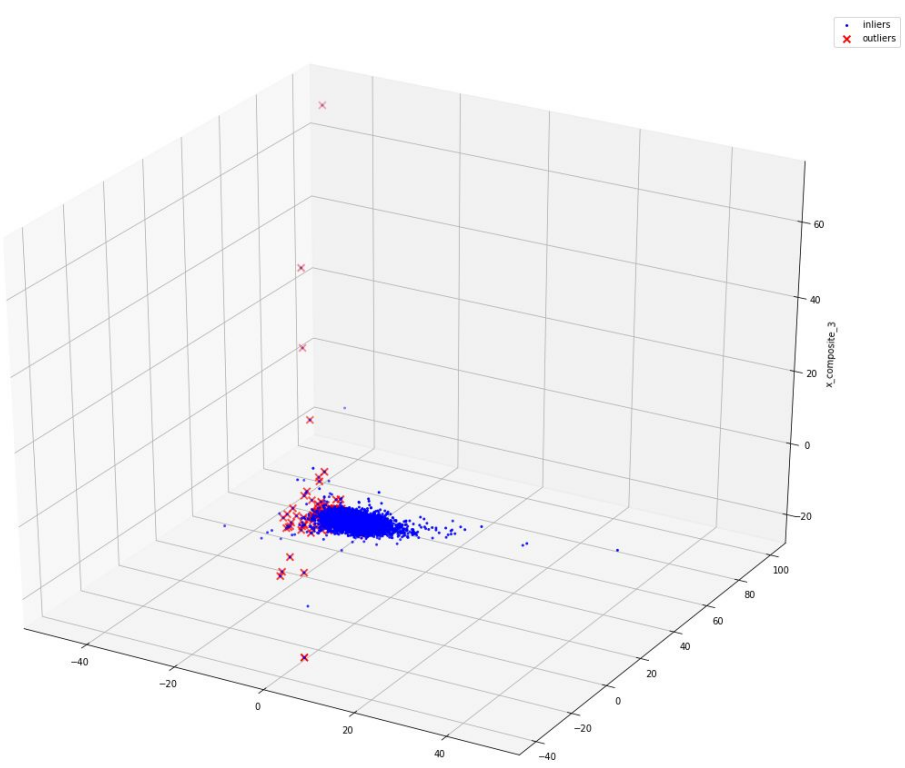
	precision	recall	f1-score	support
bankrupt_company	0.20	0.20	0.20	165
normal_company	0.97	0.97	0.97	4949
accuracy			0.95	5114
macro avg	0.58	0.59	0.59	5114
weighted avg	0.95	0.95	0.95	5114

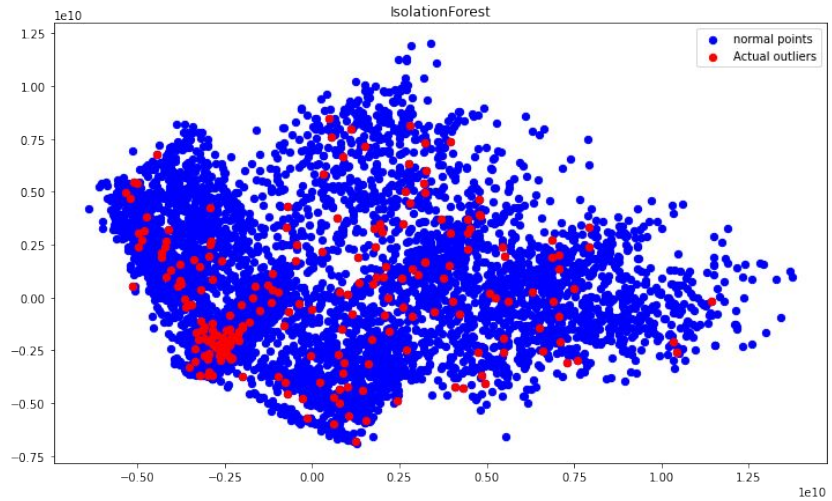
AUC: 58.6%
[[33 132]
[136 4813]]

	precision	recall	f1-score	support
bankrupt_company	0.32	0.42	0.36	55
normal_company	0.98	0.97	0.98	1650
accuracy			0.95	1705
macro avg	0.65	0.69	0.67	1705
weighted avg	0.96	0.95	0.96	1705

AUC: 69.4%
[[23 32]
[49 1601]]

EDA: Data Visualization in 2 and 3 - Dimensions using PCA



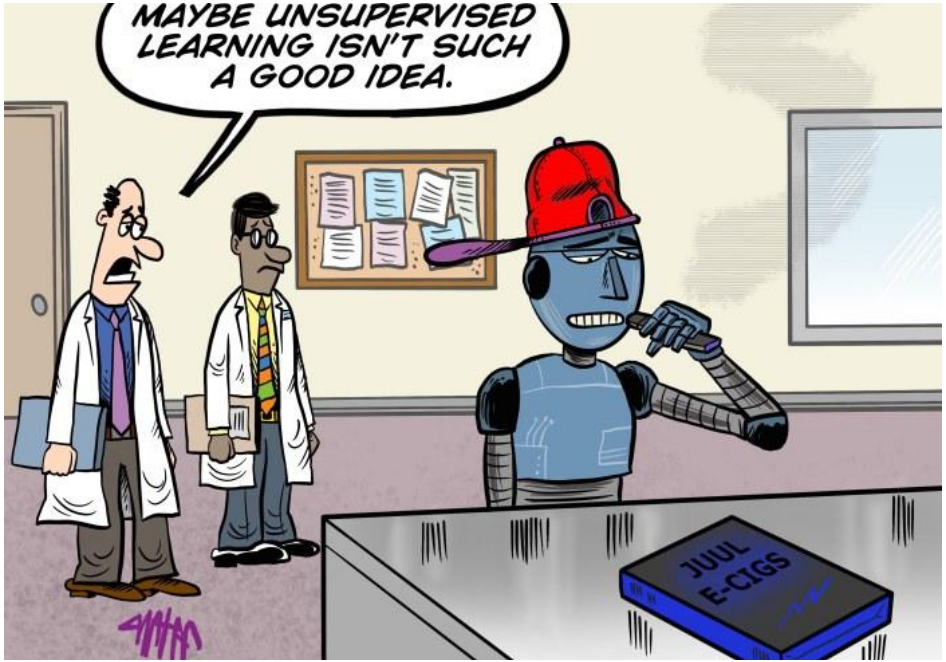


- **Classes are intertwined & overlapping. Thus, bankruptcy isn't an anomaly. Bankrupt instances, simply, happen to be sparse in the date set.**

➢ Conclusion

- The Problem at hand is not an anomaly detection problem. It is a classification problem with a highly imbalanced dataset.
- We will use different combinations of feature selection techniques, classification models and resampling techniques to reach a solution

Applying Models



Applying Model: 1. Features Selected Using VIF.

	logistic_regression	random_forest	easy_ensemble	SVM	gaussian_naive_bayes	KNN	XGBOOST
test_acc	0.863343	0.901466	0.827566	0.863343	0.886217	0.968328	0.953079
train_acc	0.884435	0.922174	0.850802	0.885413	0.902620	0.976339	0.994525
test_precision	0.177536	0.218905	0.145401	0.177536	0.173709	0.545455	0.352941
train_precision	0.206612	0.283859	0.177802	0.208852	0.210435	1.000000	0.854922
test_recall	0.890909	0.800000	0.890909	0.890909	0.672727	0.109091	0.545455
train_recall	0.909091	0.927273	1.000000	0.915152	0.733333	0.266667	1.000000
test_f1-score	0.296073	0.343750	0.250000	0.296073	0.276119	0.181818	0.428571
train_f1-score	0.336700	0.434659	0.301921	0.340090	0.327027	0.421053	0.921788
test_auc	0.921069	0.942446	0.943978	0.914380	0.914380	0.654386	0.928397
train_auc	0.959260	0.976554	0.990196	0.955605	0.955605	0.993925	1.000000

- Highest Recall on the test set is obtained from Logistic Regression, Easy Ensemble and SVM.
- Highest test precision is obtained with KNN
- XGBoost has best F1-score at 43%, on the test set.

Applying Model: 2. Features Selected Using VIF & p-values(logit-function)

	logistic_regression	random_forest	easy_ensemble	SVM	gaussian_naive_bayes	KNN	XGBOOST
test_acc	0.855132	0.891496	0.819941	0.846334	0.958358	0.971261	0.963636
train_acc	0.842002	0.895776	0.817755	0.838678	0.948964	0.976144	1.000000
test_precision	0.161972	0.199074	0.137931	0.153846	0.385714	0.666667	0.438596
train_precision	0.151679	0.228614	0.150410	0.150424	0.264706	1.000000	1.000000
test_recall	0.836364	0.781818	0.872727	0.836364	0.490909	0.218182	0.454545
train_recall	0.848485	0.939394	1.000000	0.860606	0.327273	0.260606	1.000000
test_f1-score	0.271386	0.317343	0.238213	0.259887	0.432000	0.328767	0.446429
train_f1-score	0.257353	0.367734	0.261490	0.256087	0.292683	0.413462	1.000000
test_auc	0.927592	0.920694	0.923163	0.924826	0.924826	0.704121	0.909554
train_auc	0.926745	0.967358	0.988753	0.926008	0.926008	0.993949	1.000000

- **Highest Recall on the test set is obtained from Easy Ensemble at 87.2%**
- **Highest test precision is obtained with KNN at 66%**
- **XGBoost has best F1-score at 44%, on the test set.**

Applying Model: 3. Features Selected Using VIF & l1 Regularization

	logistic_regression	random_forest	easy_ensemble	SVM	gaussian_naive_bayes	KNN	XGBOOST
test_acc	0.866276	0.921994	0.853372	0.869208	0.923754	0.967155	0.963636
train_acc	0.856668	0.913766	0.855690	0.856081	0.908291	0.975753	1.000000
test_precision	0.176030	0.273256	0.171717	0.179389	0.264151	0.428571	0.410256
train_precision	0.170534	0.262069	0.182724	0.168409	0.222628	1.000000	1.000000
test_recall	0.854545	0.854545	0.927273	0.854545	0.763636	0.054545	0.290909
train_recall	0.890909	0.921212	1.000000	0.878788	0.739394	0.248485	1.000000
test_f1-score	0.291925	0.414097	0.289773	0.296530	0.392523	0.096774	0.340426
train_f1-score	0.286271	0.408054	0.308989	0.282651	0.342216	0.398058	1.000000
test_auc	0.945355	0.954942	0.958562	0.946204	0.946204	0.690248	0.943295
train_auc	0.941090	0.969086	0.986660	0.940017	0.940017	0.991952	1.000000

- Highest Recall on the test set is obtained from Easy Ensemble at 92%
- Highest test precision is obtained from KNN att 42%
- Random Forest has best F1-score at 41%, on the test set.

Applying Model: 4. Features Selected Using p_values with OLS model.

	logistic_regression	random_forest	easy_ensemble	SVM	gaussian_naive_bayes	KNN	XGBOOST
test_acc	0.870381	0.911437	0.854545	0.853959	0.959531	0.969501	0.917889
train_acc	0.863121	0.910246	0.849433	0.846109	0.947986	0.975557	0.954634
test_precision	0.178295	0.236264	0.163763	0.160839	0.418605	0.714286	0.225806
train_precision	0.178099	0.256623	0.176471	0.164147	0.295547	1.000000	0.415617
test_recall	0.836364	0.781818	0.854545	0.836364	0.654545	0.090909	0.636364
train_recall	0.896970	0.939394	1.000000	0.921212	0.442424	0.242424	1.000000
test_f1-score	0.293930	0.362869	0.274854	0.269795	0.510638	0.161290	0.333333
train_f1-score	0.297189	0.403121	0.300000	0.278643	0.354369	0.390244	0.587189
test_auc	0.930832	0.931394	0.928540	0.926160	0.926160	0.692584	0.906975
train_auc	0.947431	0.970002	0.987577	0.944685	0.944685	0.992117	0.999433

- Highest Recall on the test set is obtained from Easy Ensemble at 85.4%
- Highest test precision is obtained from KNN att 71.4%
- Gaussian Naive Bayes has best F1-score at 51%, on the test set.

Applying Model: 5. Using Quasi Constant Method

	logistic_regression	random_forest	easy_ensemble	SVM	gaussian_naive_bayes	KNN	XGBOOST
test_acc	0.842229	0.916716	0.843402	0.842229	0.881525	0.968915	0.963636
train_acc	0.845718	0.928432	0.860774	0.846500	0.866641	0.974189	1.000000
test_precision	0.157051	0.245614	0.153595	0.152597	0.167421	0.625000	0.444444
train_precision	0.157895	0.302554	0.188141	0.158590	0.135402	1.000000	1.000000
test_recall	0.890909	0.763636	0.854545	0.854545	0.672727	0.090909	0.509091
train_recall	0.872727	0.933333	1.000000	0.872727	0.581818	0.200000	1.000000
test_f1-score	0.267030	0.371681	0.260388	0.258953	0.268116	0.158730	0.474576
train_f1-score	0.267409	0.456973	0.316699	0.268406	0.219680	0.333333	1.000000
test_auc	0.913532	0.934050	0.930705	0.911273	0.911273	0.692182	0.934391
train_auc	0.925422	0.977121	0.992948	0.925239	0.925239	0.992241	1.000000

- **Highest Recall on the test set is obtained from Logistic Regression at 89%**
- **Highest test precision is obtained from KNN at 62.5%**
- **XGBoost has best F1-score at 47.4%, on the test set.**

Applying Model: 6. Using Information Gain

	logistic_regression	random_forest	easy_ensemble	SVM	gaussian_naive_bayes	KNN	XGBOOST
test_acc	0.860997	0.907918	0.851026	0.860411	0.940176	0.970088	0.961290
train_acc	0.871334	0.922761	0.857841	0.871138	0.937036	0.976535	1.000000
test_precision	0.165441	0.222826	0.167224	0.164835	0.276190	0.666667	0.403509
train_precision	0.187579	0.284644	0.184978	0.185751	0.273775	1.000000	1.000000
test_recall	0.818182	0.745455	0.909091	0.818182	0.527273	0.145455	0.418182
train_recall	0.896970	0.921212	1.000000	0.884848	0.575758	0.272727	1.000000
test_f1-score	0.275229	0.343096	0.282486	0.274390	0.362500	0.238806	0.410714
train_f1-score	0.310273	0.434907	0.312204	0.307045	0.371094	0.428571	1.000000
test_auc	0.907174	0.929333	0.930000	0.902072	0.902072	0.717664	0.921598
train_auc	0.951172	0.971944	0.988149	0.950060	0.950060	0.992652	1.000000

- Highest Recall on the test set is obtained from Easy Ensemble at 90%
- Highest test precision is obtained from KNN at 66.5%
- XGBoost has best F1-score at 41%, on the test set.

Applying Model: 7. Using Random Forest Method

	logistic_regression	random_forest	easy_ensemble	SVM	gaussian_naive_bayes	KNN	XGBOOST
test_acc	0.853959	0.911437	0.828152	0.842815	0.938416	0.967742	0.898534
train_acc	0.855299	0.911029	0.839265	0.848651	0.933125	0.976731	0.922566
test_precision	0.172297	0.250000	0.152047	0.157556	0.291667	0.500000	0.219048
train_precision	0.169920	0.255892	0.167173	0.162791	0.258856	1.000000	0.294118
test_recall	0.927273	0.872727	0.945455	0.890909	0.636364	0.145455	0.836364
train_recall	0.896970	0.921212	1.000000	0.890909	0.575758	0.278788	1.000000
test_f1-score	0.290598	0.388664	0.261965	0.267760	0.400000	0.225352	0.347170
train_f1-score	0.285714	0.400527	0.286458	0.275281	0.357143	0.436019	0.454545
test_auc	0.939074	0.947163	0.947262	0.928127	0.928127	0.753019	0.931857
train_auc	0.938689	0.968269	0.987300	0.940101	0.940101	0.991402	0.994651

- Highest Recall on the test set is obtained from Easy Ensemble at 94%
- Highest test precision is obtained from KNN at 50.5%
- Gaussian Naive Bias has best F1-score at 40%, on the test set.

Applying Model: 8. Lasso Regularization Method

	logistic_regression	random_forest	easy_ensemble	SVM	gaussian_naive_bayes	KNN	XGBOOST
test_acc	0.862170	0.932551	0.851613	0.863343	0.916129	0.967742	0.964809
train_acc	0.844740	0.928236	0.857255	0.849628	0.916504	0.975166	0.998631
test_precision	0.164179	0.285714	0.158621	0.170370	0.161538	0.500000	0.454545
train_precision	0.153252	0.299603	0.184358	0.158371	0.188095	1.000000	0.959302
test_recall	0.800000	0.727273	0.836364	0.836364	0.381818	0.072727	0.454545
train_recall	0.842424	0.915152	1.000000	0.848485	0.478788	0.230303	1.000000
test_f1-score	0.272446	0.410256	0.266667	0.283077	0.227027	0.126984	0.454545
train_f1-score	0.259328	0.451420	0.311321	0.266921	0.270085	0.374384	0.979228
test_auc	0.894986	0.933212	0.928033	0.901972	0.901972	0.593653	0.918512
train_auc	0.920054	0.978177	0.992984	0.919903	0.919903	0.993623	1.000000

- **Highest Recall on the test set is obtained from Easy Ensemble at 83%**
- **Highest test precision is obtained from KNN at 50%**
- **XGBoost has best F1-score at 45%, on the test set.**

Applying Model: 9. Using SMOTE with Quasi Selection

	logistic_regression	random_forest	easy_ensemble	SVM	gaussian_naive_bayes	KNN	XGBOOST
test_acc	0.845161	0.897947	0.949560	0.845161	0.731965	0.919648	0.928446
train_acc	0.914064	0.968876	1.000000	0.916189	0.835777	1.000000	1.000000
test_precision	0.150502	0.203980	0.308642	0.150502	0.084711	0.211268	0.268966
train_precision	0.911788	0.958235	1.000000	0.911485	0.816539	1.000000	1.000000
test_recall	0.818182	0.745455	0.454545	0.818182	0.745455	0.545455	0.709091
train_recall	0.925986	0.983620	1.000000	0.930839	0.886552	1.000000	1.000000
test_f1-score	0.254237	0.320312	0.367647	0.254237	0.152134	0.304569	0.390000
train_f1-score	0.918832	0.970761	1.000000	0.921061	0.850107	1.000000	1.000000
test_auc	0.893278	0.904375	0.909840	0.893124	0.893124	0.763455	0.920860
train_auc	0.967303	0.994242	1.000000	0.966382	0.966382	1.000000	1.000000

- **Highest Recall on the test set is obtained from Logistic Regression & SVM at 82%**
- **Highest test precision is obtained from Easy Ensemble at 31%**
- **XGBoost has best F1-score at 39%, on the test set.**

Conclusion

- If Recall is critical to the use case, then Easy Ensemble model should be used with Random Forest feature selection, to obtain a 94% Recall value.
- If Precision is critical to the use case, then KKN model should be used with, p_values-OLS feature selection method to obtain a Precision of 71.4%
- If over-all performance is critical to the use case, then Gaussian Naive Bayes Model with, p_values-OLS feature selection method can be used to obtain F1 score of 51%.
- We can use combination of high recall models and high precision models to cross validate and obtain ideal solution.

Challenges

- Balancing the trade-off between recall and precision was a challenge.
- Feature selection and choosing a right technique.
- Exploring literature and resources to understand the problem and to find the solution was a little exhaustive.
- Deadlines felt a little strained. But it all worked out for the best.


QnA