ISyE 7406: Data Mining and Statistical Learning

Credit Card Default Prediction using Machine Learning

Prof. Yajun Mei

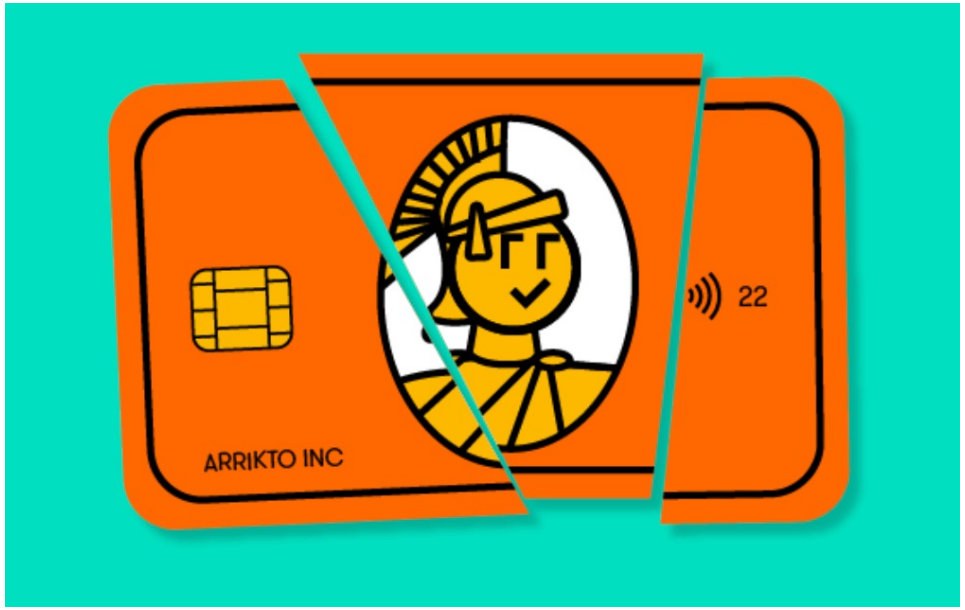
Team:

- Aman Hasan
- Akshay Amitabh
- Anant Gandhi
- Ruchit Vithani
- Siddhesh Gopanpallikar

Georgia Tech

Overview

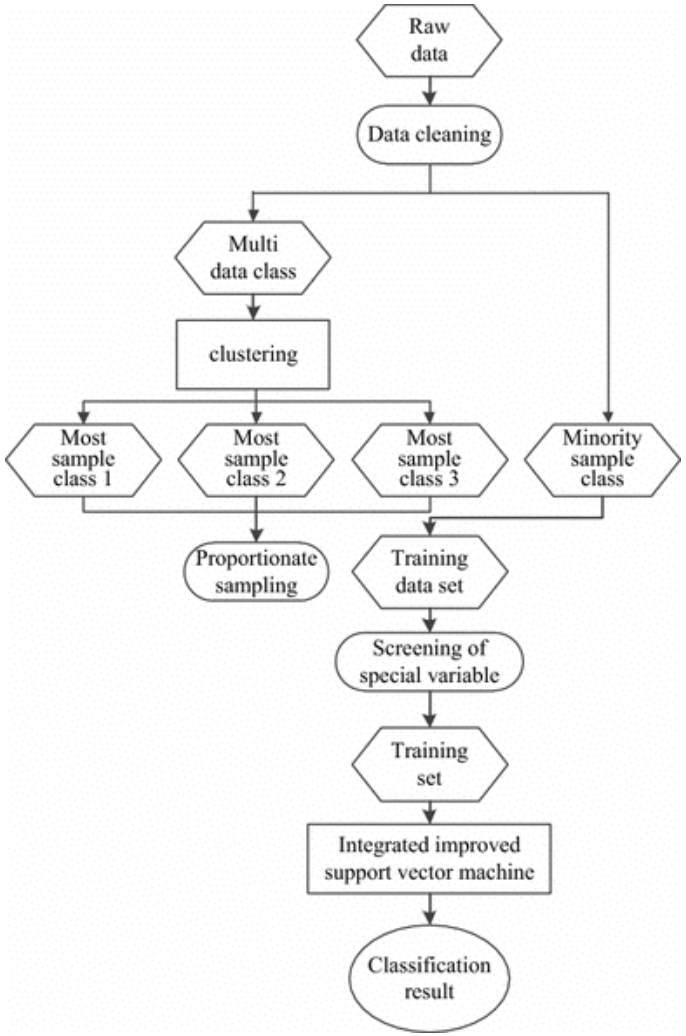
- **Objective**: In this project, we aim to develop a Machine Learning model for credit default prediction using an industrial-scale dataset provided by American Express
    - Ancillary objective is to create a better customer experience for cardholders by making it easier to be approved for a credit card
- **Dataset**: Industrial-scale dataset including time-series behavioural data and anonymized customer profile information
- **Goal**: The central problem in this project is to predict whether a customer will default on their credit card payment or not
    - The challenge is to create a model that can outperform the current model in production and improve the customer experience



Georgia Tech

Key Objectives and Plan

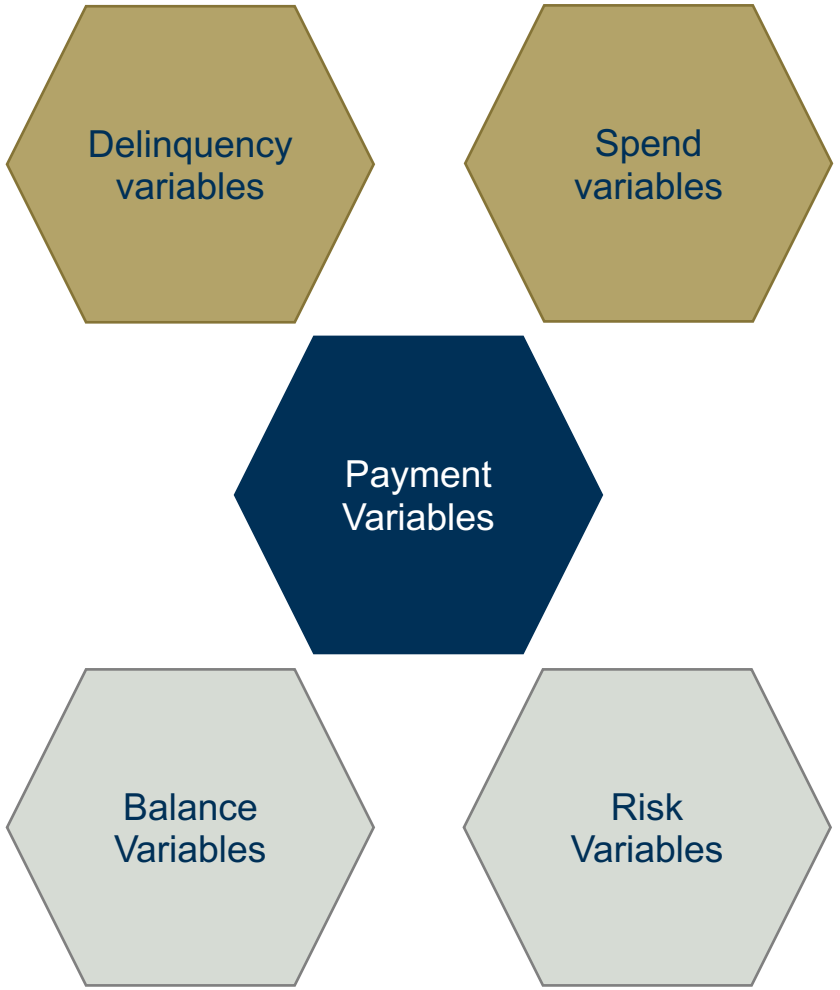
- **Preprocessing** the provided dataset by handling missing values, feature engineering, and scaling the data
- **Implementing various machine learning algorithms**, such as logistic regression, decision trees, random forests, and gradient boosting, to predict credit default
- Identifying and tuning the hyperparameters using best cross-validation performance
- Evaluating the performance of the developed models using various using a special evaluation metric
- Compare the performance of the models, develop key insights from the comparison and suggest the future work in the area



Georgia Tech

Data Description

- The dataset for this project contains **anonymized and normalized** customer profile information, including delinquency variables, spend variables, payment variables, balance variables, and risk variables
  - Actual feature names are not known for this dataset
- The features are categorized in the following **classes** :
  - D * = Delinquency variables
  - S * = Spend variables
  - P * = Payment variables
  - B * = Balance variables
  - R * = Risk variables



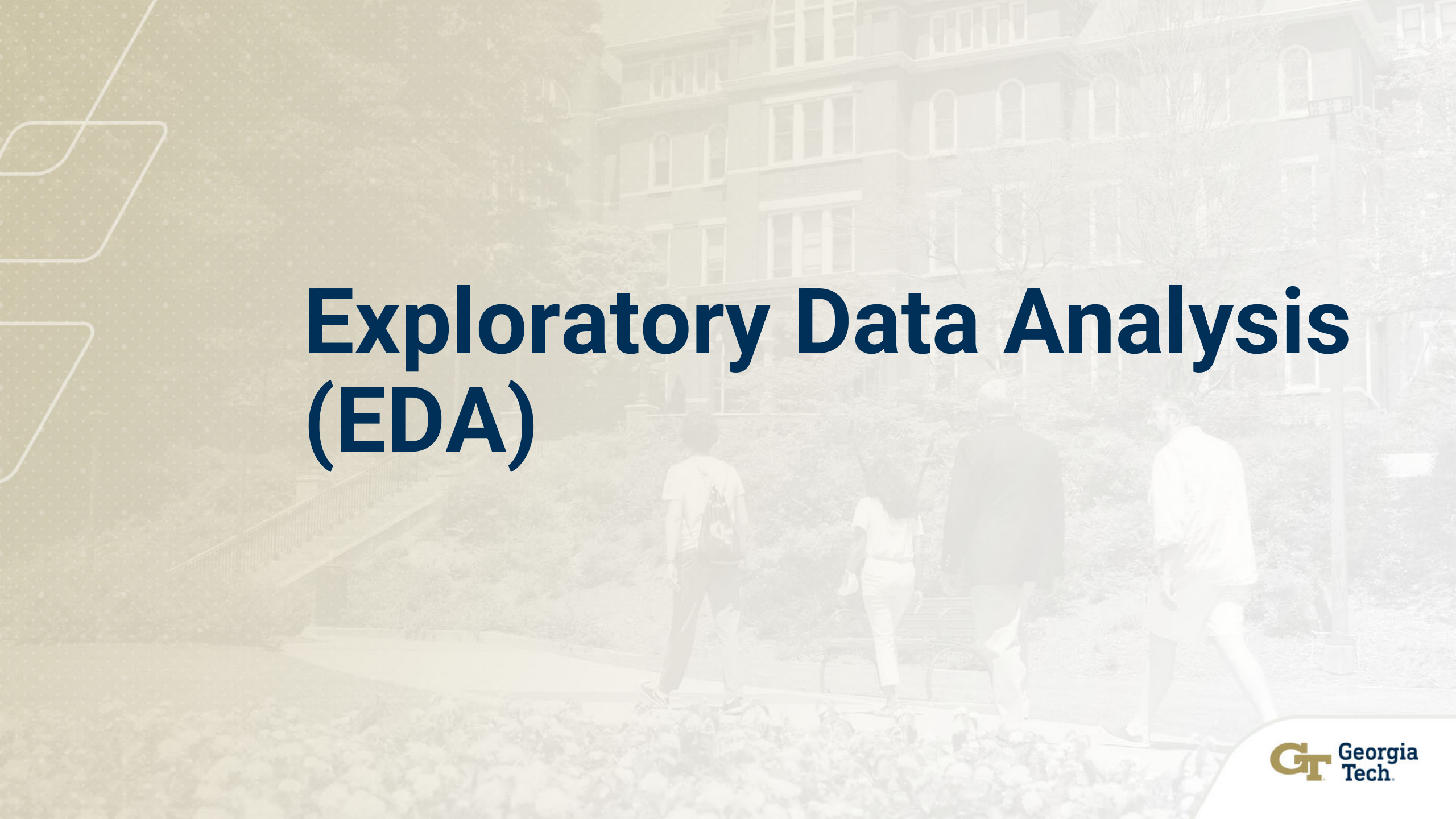
Georgia Tech

Target Variable Description

- The objective of this competition is to **predict the probability of a customer defaulting** on their credit card balance in the future
- The target variable is a **binary variable** calculated by observing the performance of the customer for 18 months after their latest credit card statement
- If the customer does not pay their due amount within 120 days of their latest statement date, it is considered a default event
- The negative class has been subsampled for this dataset at 5%, and thus receives a 20x weighting in the scoring metric

Georgia Tech

Exploratory Data Analysis (EDA)



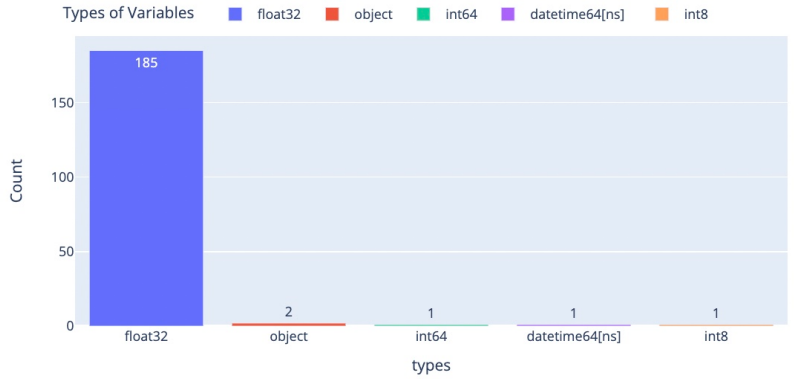
Exploratory Data Analysis

	Train Set	Test Set
# Rows	5531451	11363762
# Features	190	190
Total Customers	458913	924621
Total Defaults	118828	Don’t Know

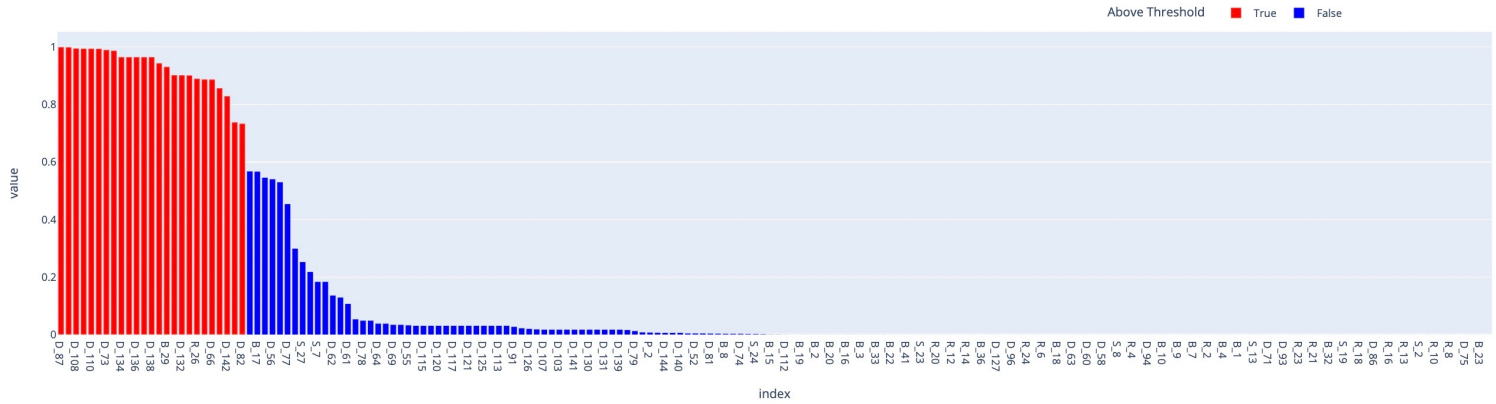
- Surprisingly, the test data size is bigger for this challenge
- We don’t know the number of defaults in test. That’s the challenge. This is what we are trying to predict

- Also, we don’t need to worry about the train-test split. The train-validation split is done during cross-validation in 80-20 ratio.

Data Type and Null Analysis



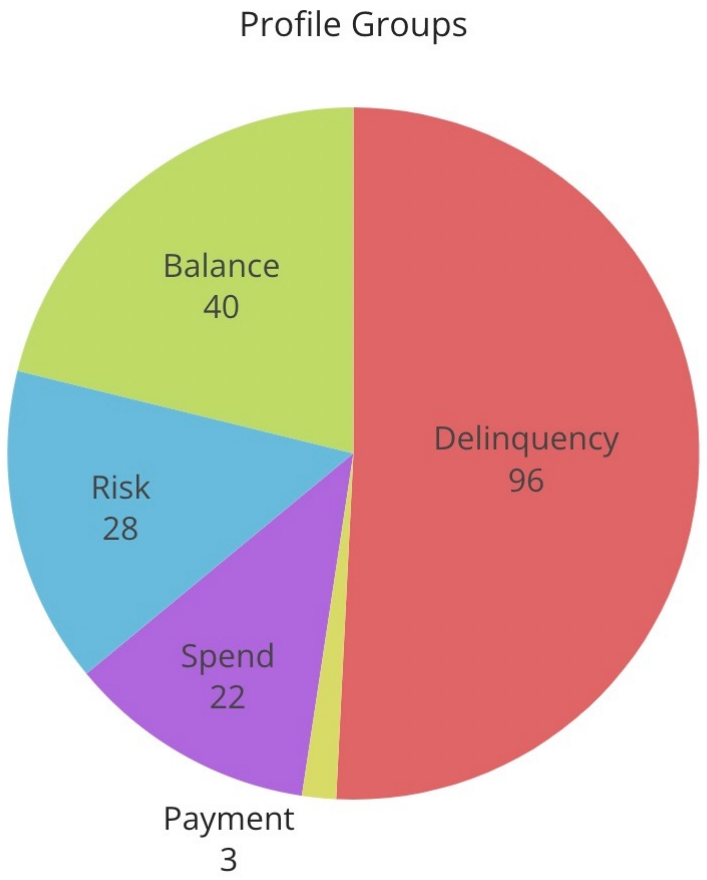
- **Datatype analysis** of variables shows that 185 variables have datatype of float followed by object and then int datatype



- **Null analysis** shows D* variables have maximum null values followed by very few null values for R* and S* variables in the dataset

Georgia Tech

Distribution of Categorical Variables

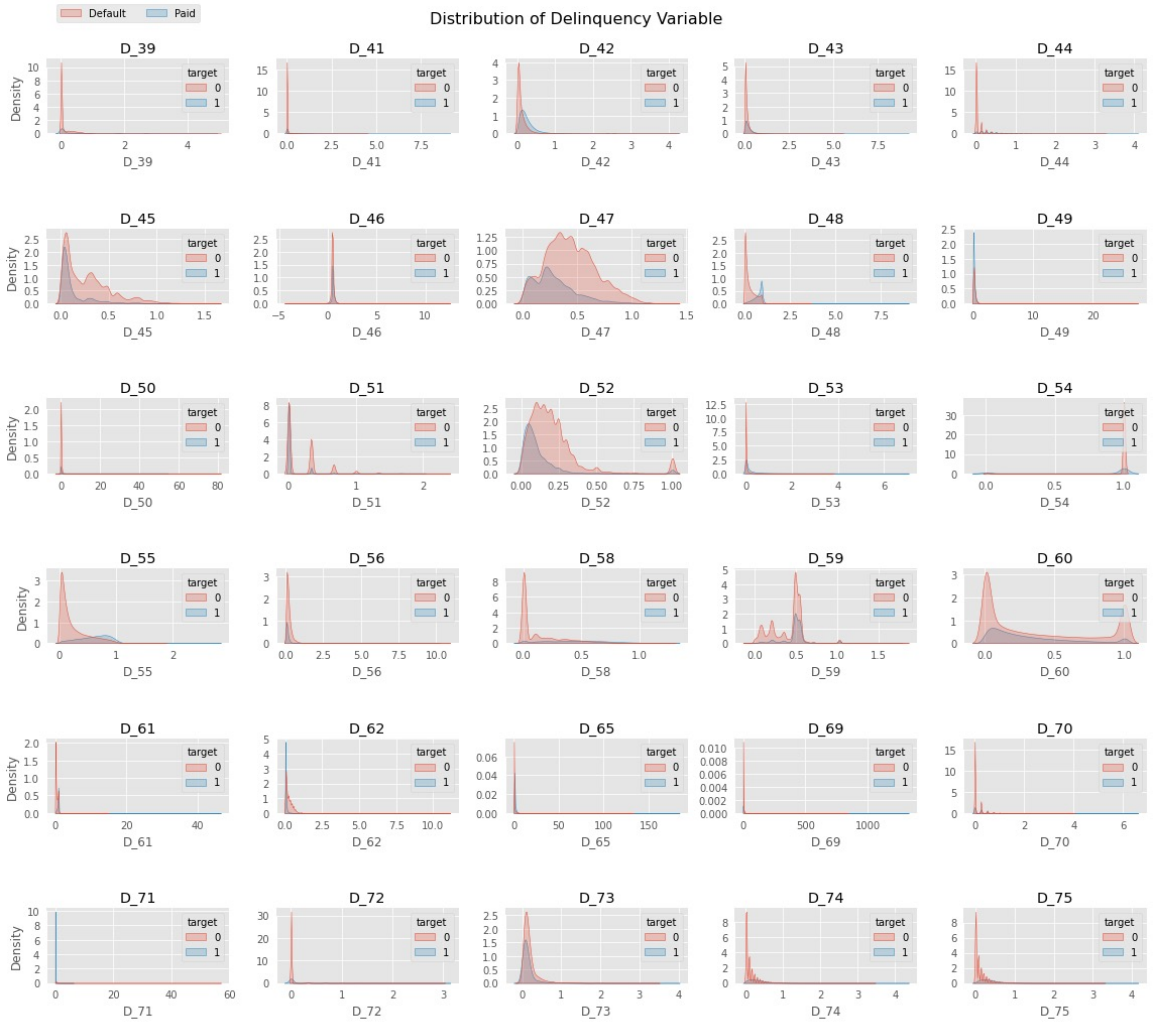


- **Maximum contribution based on count** for predicting default comes from **delinquency** variables followed by Balance and Risk variables

Georgia Tech

Univariate Analysis (1/2)

- Distribution of Delinquency Variables:
  - Most of the variables have greater density distribution for no default compared to default
  - **D_127, D_134, D_123** shows higher distribution for default than for no default distinctly suggesting these variables will have more predictive power when it comes to predicting default
  - Most variables seem to not have much predicting power basis the distributions which warrants variable selection and feature engineering
  - Most of the variables are normally distributed when it comes to default and no default, but some of these have outliers clearly visible in the plots which needs to be dealt with

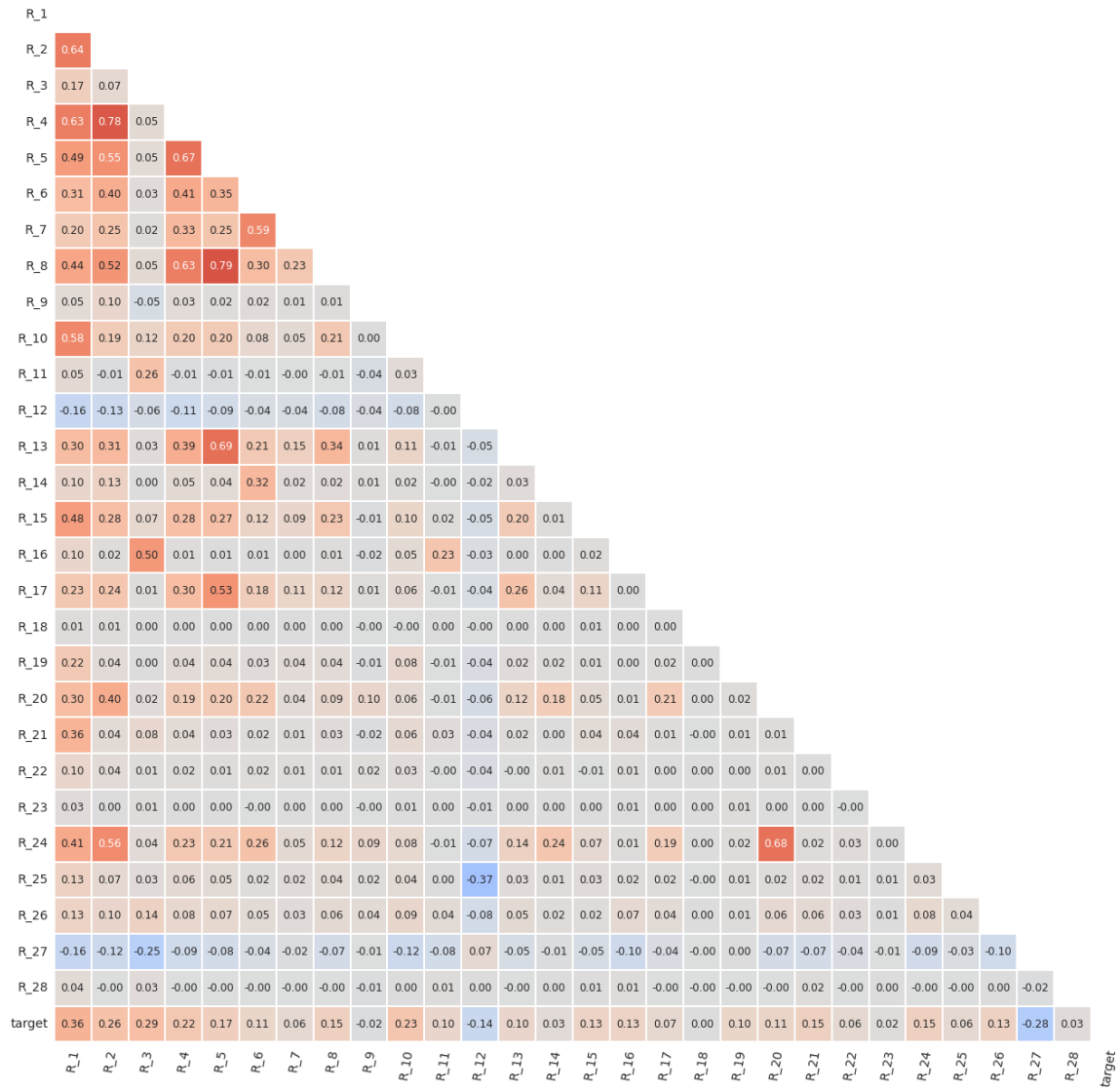


Univariate Analysis (2/2)

- Distribution of Risk and Spend Variables:
  - Most of the variables have greater density distribution for no default compared to default
  - **S_16 and S_26** shows higher distribution for default than for no default distinctly suggesting these variables will have more predictive power when it comes to predicting default
  - **R_9 and R_20** shows higher distribution for default than for no default distinctly suggesting these variables will have more predictive power when it comes to predicting default
  - Most variables seem to not have much predicting power basis the distributions which warrants variable selection and feature engineering
  - Most of the variables are normally distributed when it comes to default and no default, but some of these have outliers clearly visible in the plots which needs to be dealt with

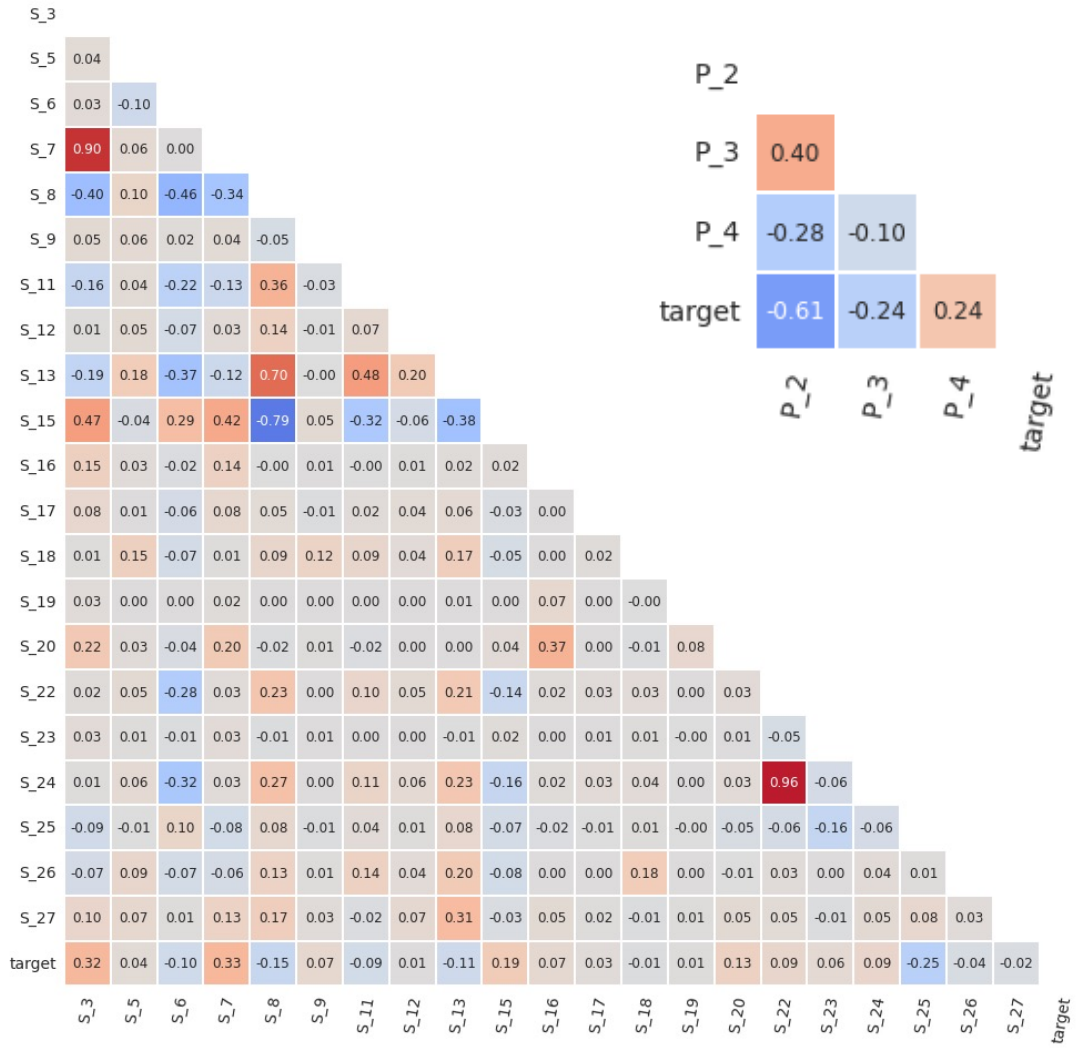
Bivariate Analysis (1/2)

- **D* Variables**:
  - D_58 and D_75 are highly correlated with 7 other D* variables considering the threshold of 0.6
  - This can lead to multicollinearity and has to be dealt with using variable selections and transformations
- **R* Variables**:
  - R_4 and R_5 are highly correlated with 4 and 3 other R* variables considering the threshold of 0.6
  - This can lead to multicollinearity and has to be dealt with using variable selections and transformations



Bivariate Analysis (2/2)

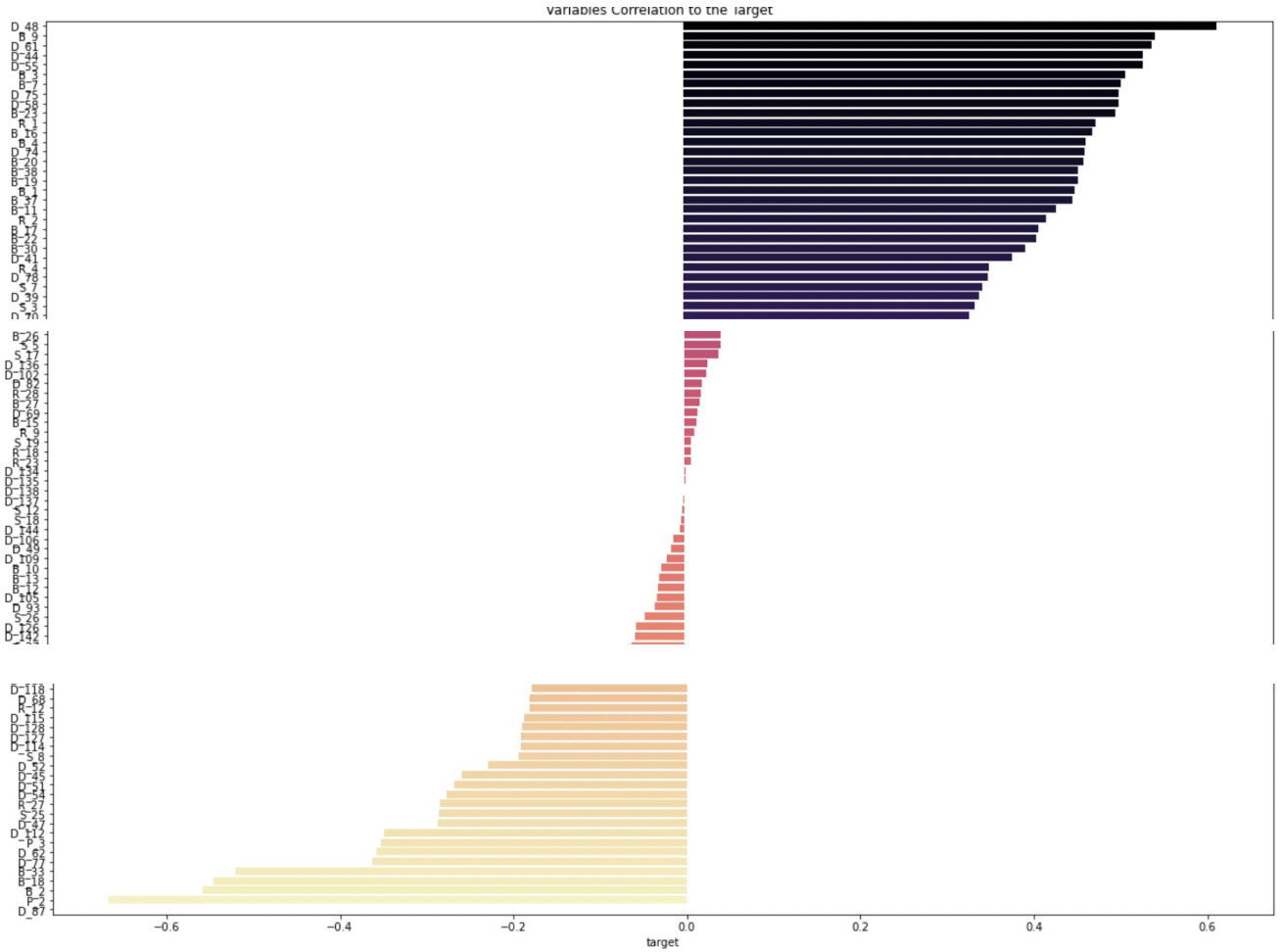
- **B* Variables**:
  - B_3, B_2 and B_20 are highly correlated with 14 other B* variables considering the threshold of 0.6
- **S* Variables**:
  - S has very less correlation with other S variables and thus multicollinearity will not be an issue here
- **P* Variables**:
  - P has very less correlation with other P variables and thus multicollinearity will not be an issue here



Georgia Tech

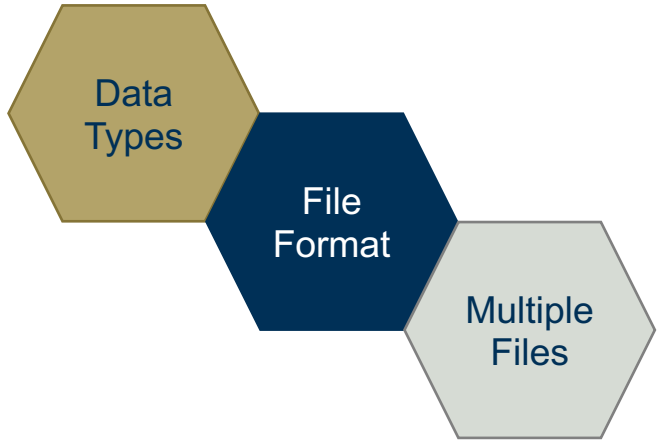
Correlation to the Target

- Correlation with the target shows **D_48, B_9, D_61, D_44, B_3 and B_7** are highly correlated with the target variable showing D* and B* variables might have higher predictive power
- **D_87, P_2, B_2 and B_18** highly negatively correlated with the target variable again showing D* and B* variables might have higher predictive power using appropriate transformations
- Most of the S* and R* variables have near to 0 correlation with the target variable showcasing very little predicting power



Reducing the data size

- This competition's tabular data is **50GB!** To engineer features from this data and train models with this data, we need to reduce data size and efficiently use memory and disk first
- **Step 1 -** Reducing Data Types
  - **Column customer_ID:** Reduce 64 bytes to 4 bytes
  - **Column S_2:** Reduce 10 bytes to 3 bytes
  - **11 Categorical Columns:** Reduce 88 bytes to 11 bytes
  - **177 Numeric Columns:** Reduce 1416 bytes to 353 bytes



- **Step 2:** Choosing the File Format
  - Data compression reduced data size without losing information. By reducing the bytes per row, the data size was compressed from **50GB to 6GB**, making it easier to store, process and improve performance
- **Step 3:** Multiple Files or Not
  - We had trouble processing the entire train and test dataset at once, so we considered **processing in chunks** and saving the data to disk as separate files

Feature Engineering

Feature Engineering

	customer_ID	S_2	P_2	D_39	B_1	B_2	R_1	S_3	D_41	B_3	...	D_136	D_137	D_138	D_139	D_140	D_141	D_14
0	-4532153018459703766	2017-03-09	0.938469	0	0.008724	1.006838	0.009228	0.124035	0.0	0.004709	...	-1	-1	-1	0	0	0.0	-127
1	-4532153018459703766	2017-04-07	0.936665	0	0.004923	1.000653	0.006151	0.126750	0.0	0.002714	...	-1	-1	-1	0	0	0.0	-127
2	-4532153018459703766	2017-05-28	0.954180	3	0.021655	1.009672	0.006815	0.123977	0.0	0.009423	...	-1	-1	-1	0	0	0.0	-127
3	-4532153018459703766	2017-06-13	0.960384	0	0.013683	1.002700	0.001373	0.117169	0.0	0.005531	...	-1	-1	-1	0	0	0.0	-127
4	-4532153018459703766	2017-07-16	0.947248	0	0.015193	1.000727	0.007605	0.117325	0.0	0.009312	...	-1	-1	-1	0	0	0.0	-127

5 rows × 190 columns

Georgia Tech

Feature Engineering

- Our dataset contains 190 features. However, they alone are not sufficient
- We create customer level aggregate features before we can feed them in any ML algorithm
- Continuous Variable Feature Engineering:
  - Aggregate on customer level and compute:
    - Mean
    - Standard Deviation
    - Minimum
    - Maximum
    - Last Value

Feature Engineering

- Our dataset contains 190 features. However, they alone are not sufficient
- We create customer level aggregate features before we can feed them in any ML algorithm
- Categorical Variable Feature Engineering:
  - Aggregate on customer level and compute:
    - Count
    - Last Value
    - Number of Unique Classes

Feature Engineering

- After Feature Engineering:

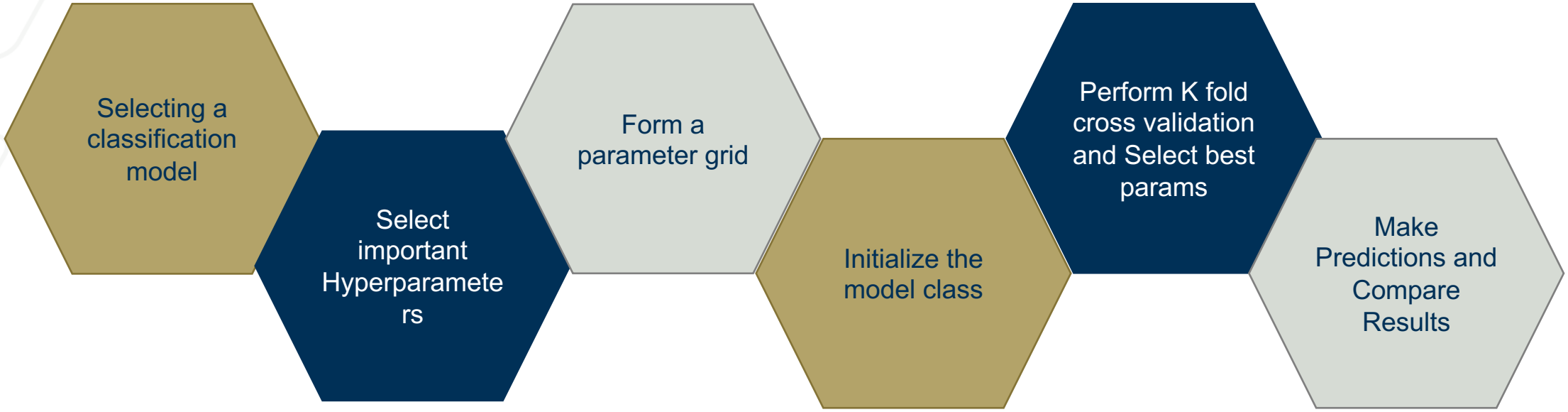
customer_ID	P_2_mean	P_2_std	P_2_min	P_2_max	P_2_last	D_39_mean	D_39_std	D_39_min	D_39_max	D_39_last	...	D_63_nunique	D_6
-9223358381327749917	0.415868	0.057145359	0.340178	0.498727	0.387708	2.615385	4.628506986	0	16	0	...	1	
-9223193039457028513	0.974068	0.013093883	0.964483	1.002478	1.001372	0.000000	0.0	0	0	0	...	2	
-9223189665817919541	0.802447	0.038025034	0.694073	0.828761	0.694073	0.000000	0.0	0	0	0	...	1	
-9223188534444851899	0.791203	0.002687729	0.786647	0.794826	0.787945	0.000000	0.0	0	0	0	...	1	
-9223173911659837606	0.115666	0.078554217	0.038207	0.252421	0.040486	4.384615	6.144624501	0	17	13	...	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	
9223275399009481284	0.508722	0.034856297	0.446763	0.580708	0.502093	3.769231	8.427671151	0	31	8	...	1	
9223300670094807586	0.128279	0.138330718	-0.035435	0.395510	0.118304	5.461538	6.213096177	1	22	6	...	1	
9223303087902649707	0.364807	0.074824325	0.271125	0.517709	0.368901	3.692308	8.576951822	0	23	0	...	1	
9223345210145379887	0.848886	0.059538209	0.776463	0.940112	0.902587	4.000000	7.118052168	0	23	0	...	1	
9223350112805974911	0.232830	0.064644629	0.102516	0.305949	0.305828	8.538462	11.73041193	0	33	31	...	1	

458913 rows × 918 columns

Georgia Tech

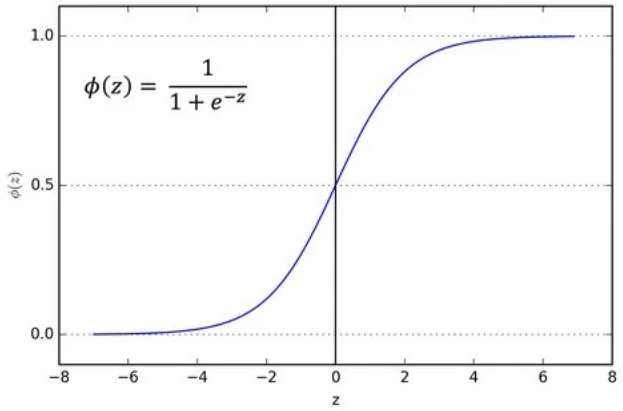
Model Development

Modelling Steps

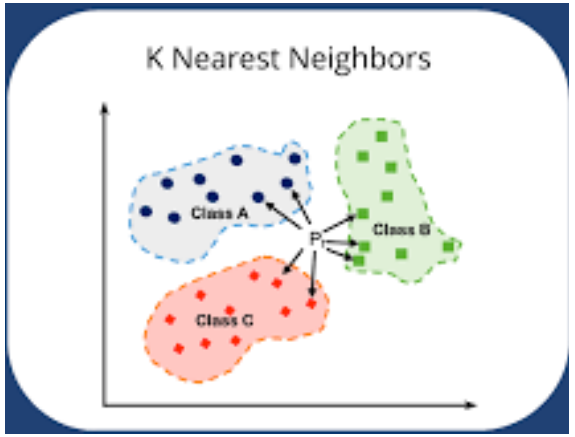


Georgia Tech

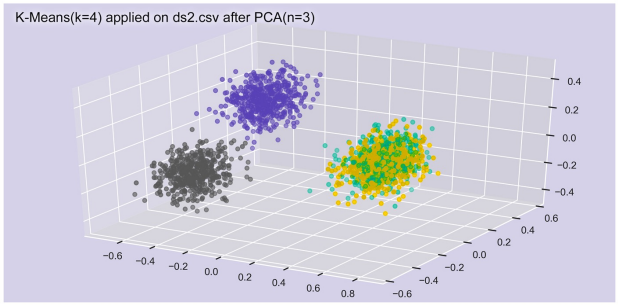
Models Used



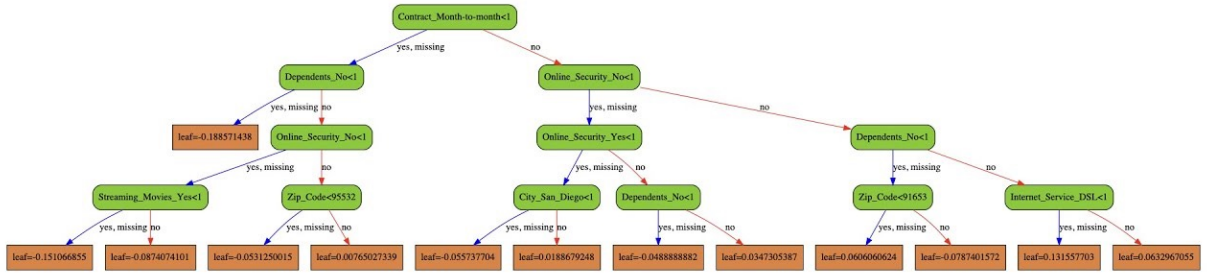
Logistic Regression



KNN



PCA + XGBoost

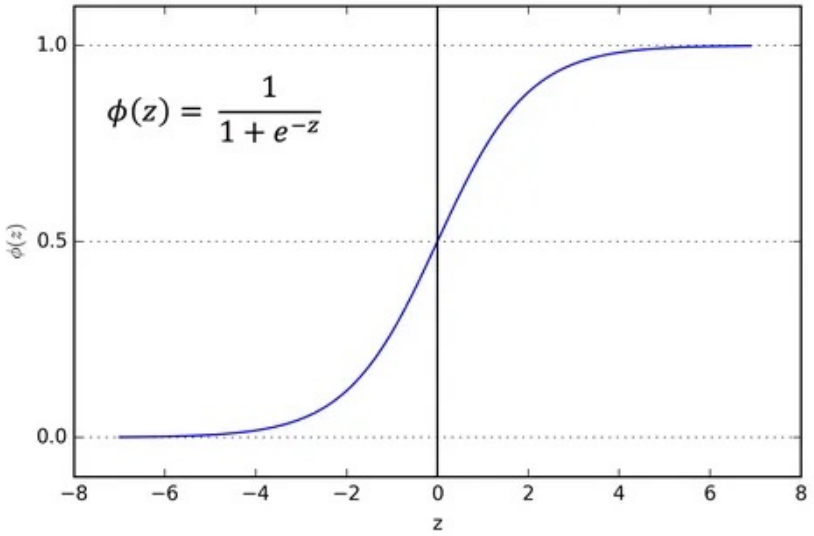


XGBoost

Georgia Tech

Model 1: Logistic Regression

- For logistic regression, there are no major hyperparameters.
- However, as there are huge number of predictor variables (~ 1000 features), we regularize the logistic regression using L1 penalty (LASSO)
- Lambda range: {0.01, 0.1, 1}
- Our cross-validation selects 0.01 as the best value of the regularization parameter lambda
- The fitted model is then used to make predictions on the test set

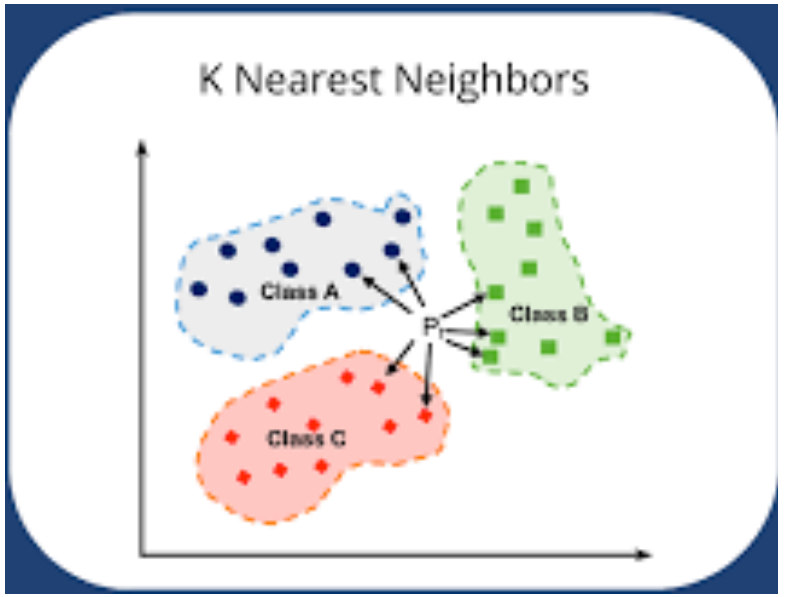


https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/

Georgia Tech

Model 2: KNN Classifier

- The KNN algorithm observes data points in high-dimensional space and classifies the data points according to the majority vote labels of the K Nearest neighbors
- KNN model is very compute intensive.
- Let’s analyze the complexity of the algorithm
- For each data points, we find the K closest neighbors
  - The closeness is computed using distance metric on d-dimensional vectors.



http://primo.ai/index.php?title=K-Nearest_Neighbors_%28KNN%29

Georgia Tech

Model 2: KNN Classifier

- Complexity: O(n * d) for each query in the algorithm
- For our case: O(500000 * 1000) = **Impossible!!!**
- Therefore, this algorithm is **super-slow** and could not converge on our machine

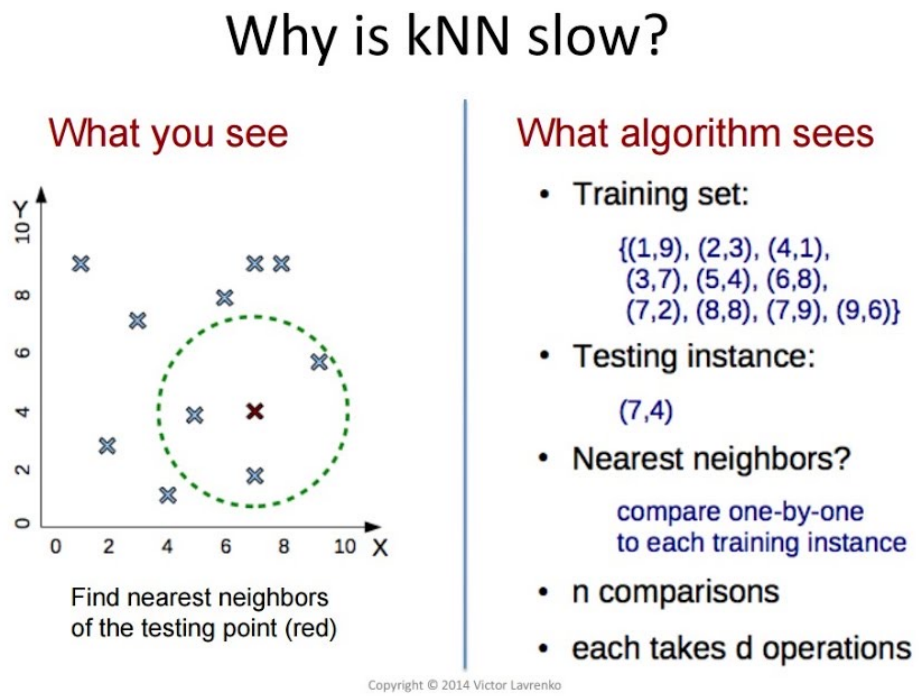
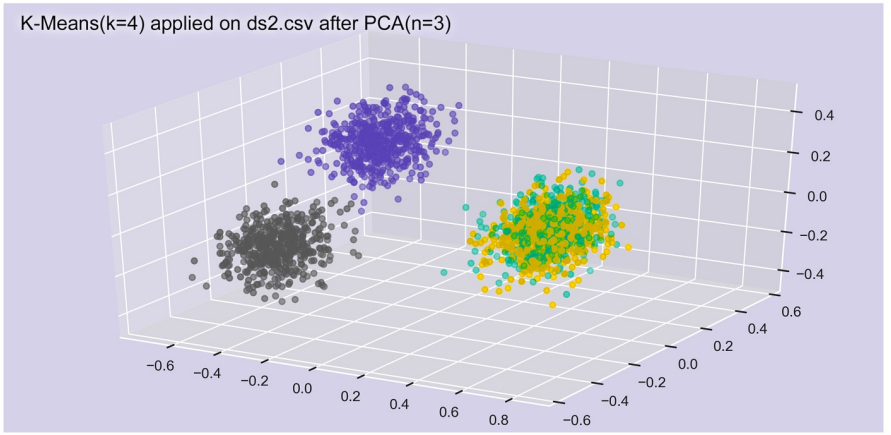


Image Reference: https://youtu.be/UPAnUE_g5SQ

Model 3: PCA + XGBoost

- Data dimensionality ~ (500000, 1000)
- The best first approach is to try reducing the dimensionality to make computation process easy
- Perform PCA to reduce dimension to (500000, 100)
  - Select 100 most significant components
  - 100 was selected on the basis of cross validation performance among (30, 100, 300)
- Number of features is still 100
  - Tree based algorithms are best choice of classification algorithms for this kinds of data
  - Train random forest classifier on the significant components



https://blog.bioturing.com/2020/10/12/the-why-when-and-how-of-3d-pca/

Model 4: XGBoost

- Generally, ensemble learning algorithms do well on this kinds of tasks
  - Where the number of features are huge
  - And the task is to classify data into pre-defined set of classes
  - **Reason**: Input from multiple models reduces. Weak learners help reduce the bias of the model created due to large feature space

- We try XGBoost classifier with the following parameters
  - Max Depth: 5
  - Learning Rate: 0.05
  - Subsample Rate: 0.75
  - Cosample By Tree: 0.6

Evaluation Metric

- Evaluation metric is the mean of two metrics:

$$M = 0.5 \cdot (G + D)$$

  - Where, G = Normalized Gini Coefficient
  - And D = Default rate captured at 4%

- **G**: Measures how well the model is able to rank the instances in order of their predicted probabilities of default
- **D**: Measures how well the model is able to capture the positive instances (defaults) within the top 4% of the predicted probabilities

- Goal: Maximize the metric M

Results

- We submit the results on the test set on the Kaggle to see the performance on the test set

Model	Train Performance	Test Performance
Logistic Regression	0.2215	0.1804
KNN	NA	NA
PCA + XGBoost	0.5899	0.5558
XGBoost	**0.7990**	**0.7934**

- XGBoost on all set of features performs better than all models
- Rank 1 on Kaggle is ~ 0.81

Georgia Tech

Discussion

- Hyperparameter tuning using cross-validation is extremely important process
  - More than 1000 positions on the leaderboard has score of 0.80.
  - Thus, it becomes extremely competitive to improve rank
  - Better model choice, smart feature engineering, necessary data cleaning and preprocessing proper parameter tuning are extremely important for better model performance
    - Compute resources (GPU power, more cores) are important, but secondary
- Data Cleaning, Preprocessing and Feature Engineering is 95% job in the project
  - Reducing the size of data from 50GB to 5GB was the best part in the project.
- Random Forest algorithm and it’s derivatives are the most used and recommended algorithms for the classification problem in Kaggle competitions

Future Ideas

- The standard Machine Learning models are explainable at times, but, they lack in capturing certain important information from the data. For example,
  - Time Series Trends
  - Latent interactions among features
  - Heavy reliance on Feature Engineering
  - They can learn, but they can’t be smart. They can’t be creative.
    - E.g. Once the model is trained, it will give same output for the given input.
    - ChatGPT can generate different responses to the same question

Future Ideas

- For this task, one can explore Deep Learning models like Convolutional Neural Network to capture customer behavior patterns in the time series.
- One can also try Sequence Encoders like LSTM models, that can encode the sequence into useful features.
- Limitations:
  - These models are useful as they help in getting rid of manual feature engineering
  - However, they are not interpretable

Conclusions

- In this project, we tried to solve the complex problem of predicting the default events using Machine Learning algorithms
- We demonstrated the important process of data engineering, data cleaning, preprocessing and feature engineering
- Trained classification algorithms, fine-tuned the model parameters using cross-validation and compared the algorithms on the basis of their performance