TEAM 1

8:30-8:42

Machine Learning: Quality Prediction of a Mining Operation

Federico Aguilar Castellanos
Michel Chamoun

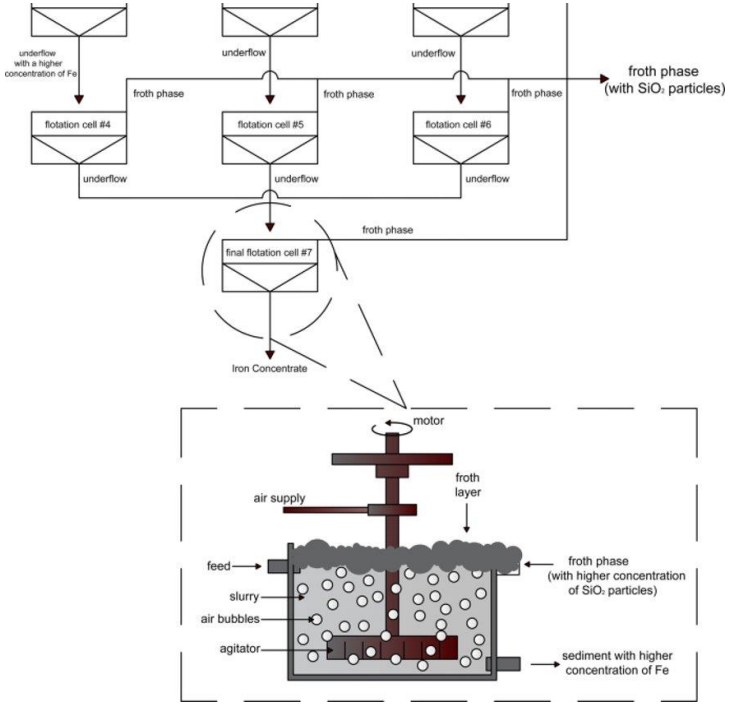
December 6th, 2021

Background Information

Ores (i.e., mined rocks) do not naturally occur as single valuable mineral, like iron or gold; instead, they are mixtures containing unwanted rocks or minerals, such as silica.

The valuable mineral (iron) is separated from the **impurity mineral** (silica) by froth flotation.

Froth Flotation is the process of physically separating particles based in their ability to adhere to air bubbles. Specifically **iron minerals** remain in the water, while **silica particles** attach to the bubbles and float to the surface. The efficiency of the process is measured by the purity of the flotation concentrate.



Pu, Y., Szmigiel, A. & Apel, D.B. Purities prediction in a manufacturing froth flotation plant: the deep learning techniques. *Neural Comput & Applic* **32,** 13639–13649 (2020). https://doi.org/10.1007/s00521-020-04773-2

Task and Dataset

Objective: Apply ML techniques to forecast the percent of silica in the flotation concentrate.

- Evaluate the effect of measurement interpolation of sequential data.
- Asses the performance of the ML models under a smaller lookback and forecast window (20-sec window vs. 1 and 6-hr window).

Type of task: Supervised Learning

Dataset: Sensor measurements sampled every 20 seconds during a 6-month period from March 2017 to September 2017.

- Total of 737,453 entries:
- Timestamp of measurements
- Process input variables
- Process equipment variables
- Process output variables
- **Label: % Silica Concentrate**

```
RangeIndex: 737453 entries, 0 to 737452
Data columns (total 24 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   date                        737453 non-null  datetime64[ns]
 1   % Iron Feed                 737453 non-null  float64
 2   % Silica Feed               737453 non-null  float64
 3   Starch Flow                 737453 non-null  float64
 4   Amina Flow                  737453 non-null  float64
 5   Ore Pulp Flow               737453 non-null  float64
 6   Ore Pulp pH                 737453 non-null  float64
 7   Ore Pulp Density            737453 non-null  float64
 8   Flotation Column 01 Air Flow  737453 non-null  float64
 9   Flotation Column 02 Air Flow  737453 non-null  float64
 10  Flotation Column 03 Air Flow  737453 non-null  float64
 11  Flotation Column 04 Air Flow  737453 non-null  float64
 12  Flotation Column 05 Air Flow  737453 non-null  float64
 13  Flotation Column 06 Air Flow  737453 non-null  float64
 14  Flotation Column 07 Air Flow  737453 non-null  float64
 15  Flotation Column 01 Level   737453 non-null  float64
 16  Flotation Column 02 Level   737453 non-null  float64
 17  Flotation Column 03 Level   737453 non-null  float64
 18  Flotation Column 04 Level   737453 non-null  float64
 19  Flotation Column 05 Level   737453 non-null  float64
 20  Flotation Column 06 Level   737453 non-null  float64
 21  Flotation Column 07 Level   737453 non-null  float64
 22  % Iron Concentrate          737453 non-null  float64
 23  % Silica Concentrate        737453 non-null  float64
dtypes: datetime64[ns](1), float64(23)
```

```
count    737453.000000
mean          2.326763
std           1.125554
min           0.600000
25%           1.440000
50%           2.000000
75%           3.010000
max           5.530000
Name: % Silica Concentrate, dtype: float64
```

Feature Analysis

Challenge: Sensor timestamps of the original data did not reflect the 20 sec sampling frequency – as the 180 measurements per hour were stored under the same hourly timestamp label.

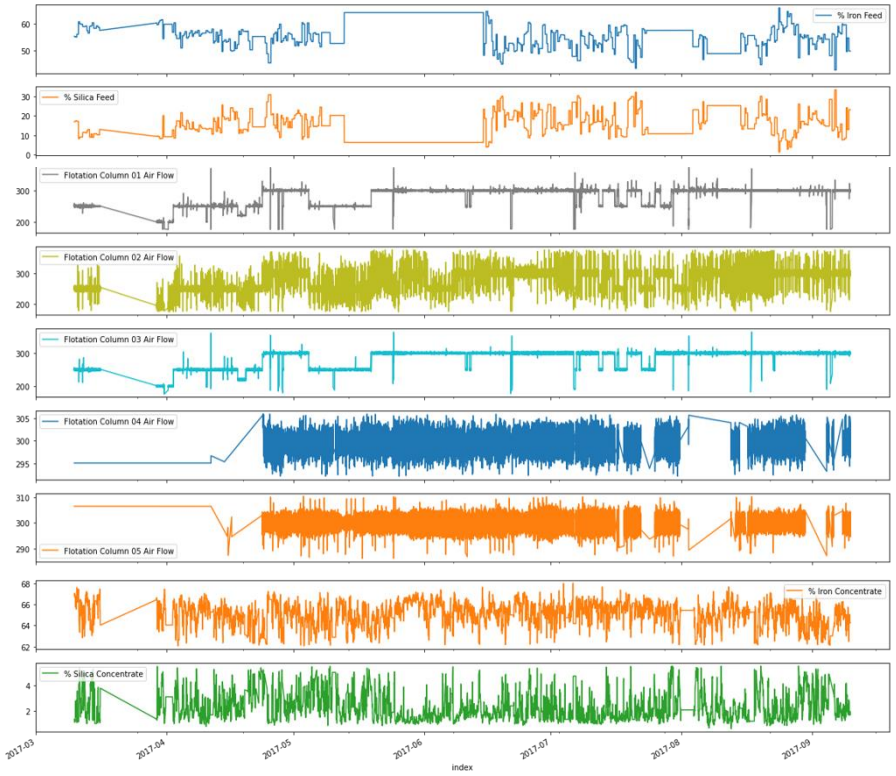
- Reindexing of the data creating a Pandas date-range
- Matching the 180 measurement recordings to new index

Challenge: Regions of linear interpolation were observed for all the variables in the dataset.

- The period between April 2017 to May 2017 showed the least amount of linear interpolation

Challenge: The % Silica Feed was strongly correlated to the % Iron Feed, which is to be expected as these are introduced simultaneously in the process. Additionally, there was a correlation between the process variables of the different flotation columns.

- Collinear data points were removed to prevent model overfitting
- The variable % Silica Feed was dropped from the data
- Only the process variables for the Flotation Column 02, Flotation Column 05, and Flotation Column 07 were kept for the model



Approach

ML models considered:
Multilayered Perceptron (MLP) Neural Network
Random Forest Regression – with and without time-step consideration
Long-Short Term Memory (LSTM) Neural Network

Objective:
Compare the forecasting performance of different ML models with varying degrees of complexity
- Random Forest vs. Neural Networks
- Univariate LSTM vs. Multivariate LSTM

Asses the effect of the time-dependency of the measurements on the generated predictions
- MLP vs. LSTM
- Random Forest Regression with time consideration vs. without time consideration

Time-Series Approach: The time-series data was transformed into input samples of past observations and output samples of future observations. Explicitly, the lagged observations (t-1) of the input variables were used to forecast the % Silica Concentrate at time (t).

Regression Approach: The data is treated as observations not taking into consideration any notion of time nor the effect of past observations.

Evaluation Setup

What is the baseline? The baseline performance for the time-series models were their less complex (regression) counterparts, which did not account for the time-dependency of the dataset.

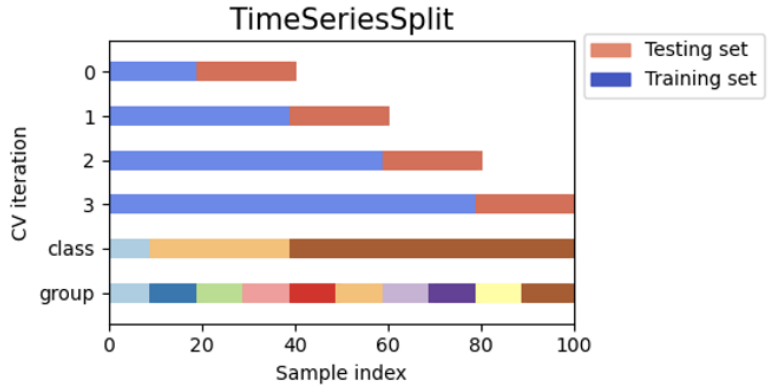
Splitting of the data

Regression models: train – test split by scikit-learn machine learning library.

- 70% training data was used to fit the models
- 20% validation data used for hyperparameter tunning
- 10% test data used to make predictions and report the performance of the model

Time-Series models: train – test split into chronological sets (by indexing) to check parameter stability of the model over time.

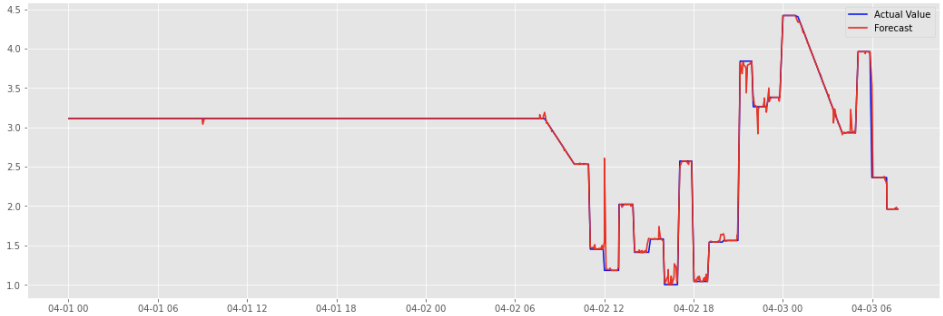
- Train/test/validation split (60%/20%/20%)
- Non-random split to maintain the time dependency of the measurements
- **Random Forest Cross-Validation -** TimeSeriesSplit by scikit-learn machine learning library setting the number of splits to 3 for cross-validation.
- **LSTM Cross-Validation:** GridSearchCV scikit-learn



Results – Regression Models

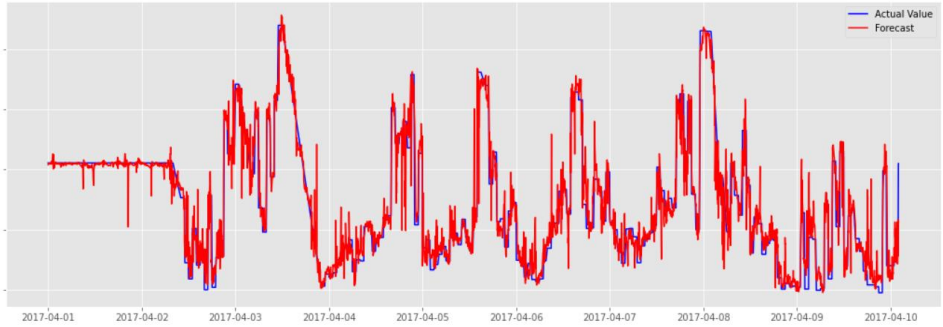
RANDOM FOREST

Mean squared error: 0.047
Root Mean squared error: 0.2168



MLP

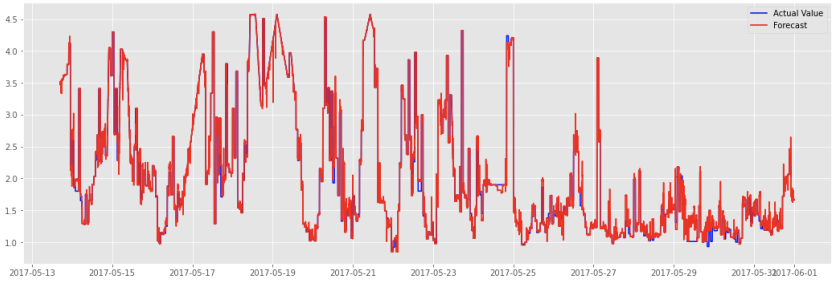
Mean Squared Error: 0.1888
Root Mean squared error: 0.4264



Results – Time Series Models

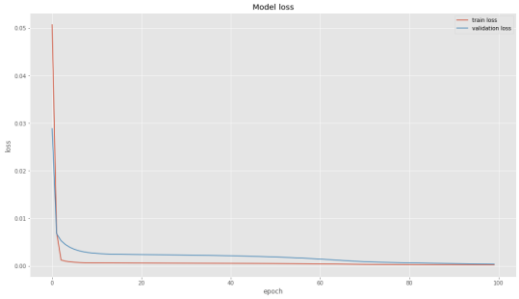
RANDOM FOREST FORECAST EVALUATION

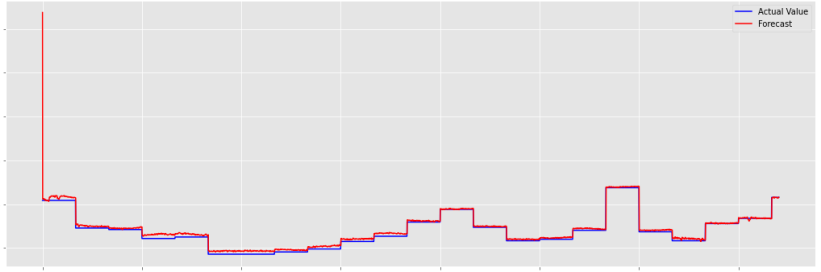
Mean absolute error : 0.0398
Root Mean squared error: 0.1039



MULTIVARIATE LSTM MODEL EVALUATION

Mean absolute error : 0.10896
Root Mean squared error: 0.1685





Conclusion

- Time-series models performed better than the regression models at predicting the % Silica Concentrate.
- Random Forest with time series consideration performed slightly better than the LSTM neural network.
- Random Forest regression algorithm performed better than the MLP neural network.
- The better performance of the Random Forest models could be attributed to the susceptibility of neural networks to hyperparameter tunning, which was evidenced during the cross-validation of these models.
- Most importantly, all the considered models converged; thus, the objective to predict the percentage of silica in the concentrate using a 20-second time step was met with great accuracy.

TEAM 2

8:42-8:54

Ulabox customer segmentation

Team members
Kiana Sharifpour
Hanie Ghalambor
Soroosh Sabaei

Goal of the Project

Business Goals?

- How Do Customers Shop Online
- Explore Online Shopping Behavior

Technical Goal?

- How to determine the number of clusters?

Task and Dataset

What is the task?

- Unsupervised Learning

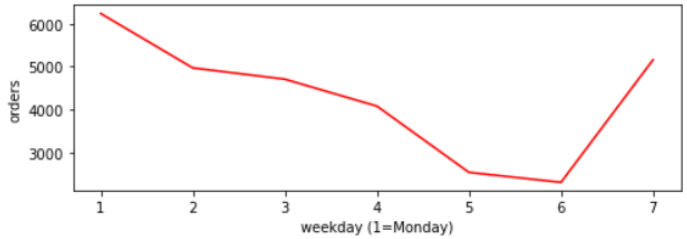
Statistics Related to the dataset?

- The number of customers: 10239
- The number of orders: 30000
- The products are related to: Food, Fresh, Drinks, Home, Beauty, Health, Baby, and pets

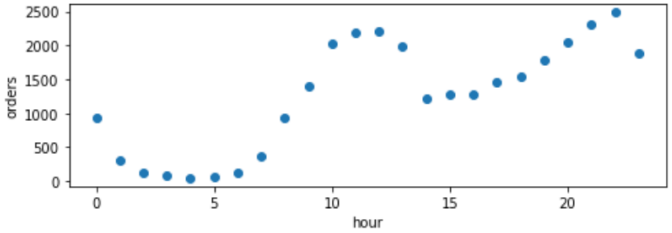
Approach: Road map of the task

1) Overview of data:

Frequency of order (week)



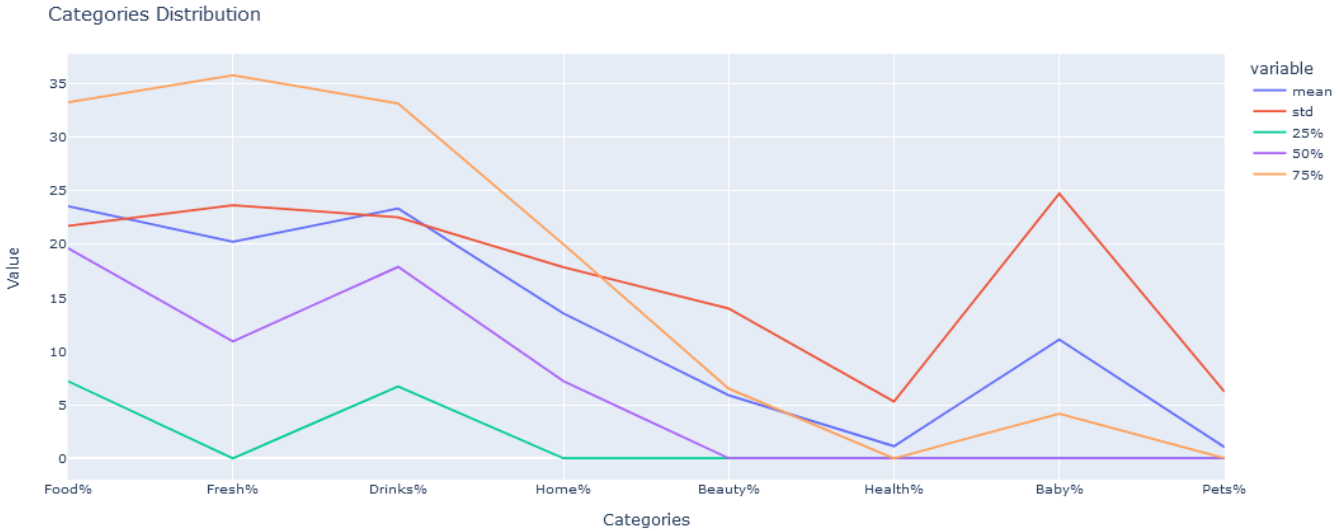
Frequency of order (day)



Customer Heatmap

Approach: Road map of the task

1) Overview of data:

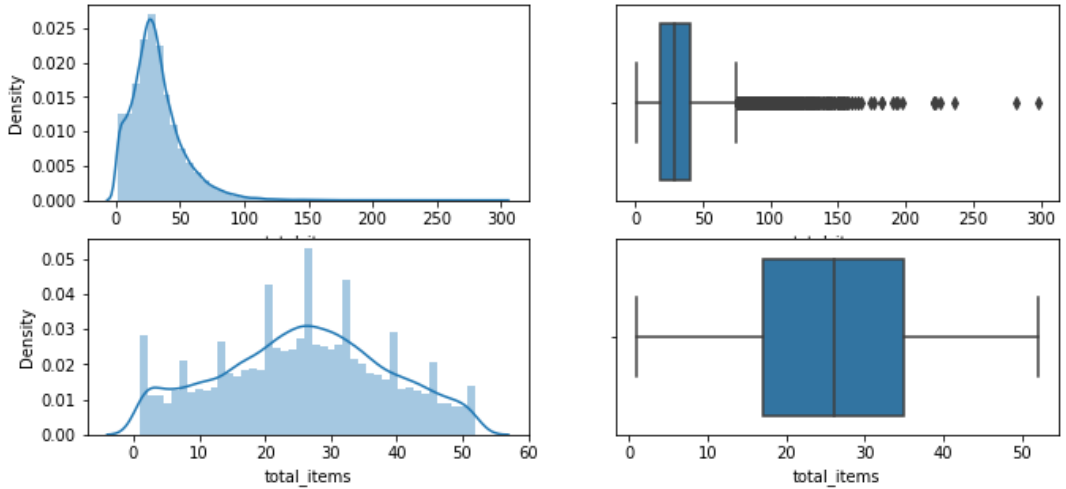


Approach: Road map of the task

2) Outlier detection:

- In the most general sense, an *outlier* is a data point which differs significantly from other observations.

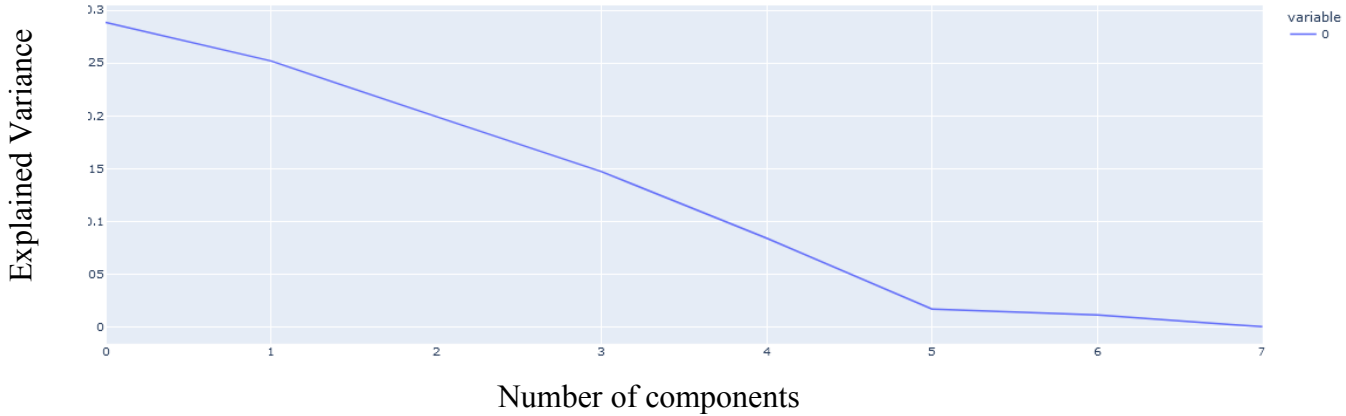
IQR = Q3 – Q1
Lower Bound = (Q1 – 1.5*IQR)
Upper Bound = (Q3 + 1.5*IQR)



Approach: Road map of the task

3) Dimentionality Reduction: **Principal Component Analysis** (PCA)

- PCA reduces the dimensions with the lowest ability to explain variances
- First, scaling the data
- Choose the number of components which explain most of the variances
- Using PCA, the columns lose their meanings



Approach: Road map of the task

4) Clustering Approach: 1) K-means

- It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum.

- The optimal number of clusters:

Approach: Road map of the task

4) Clustering Approach: 1) K-means

- It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum.

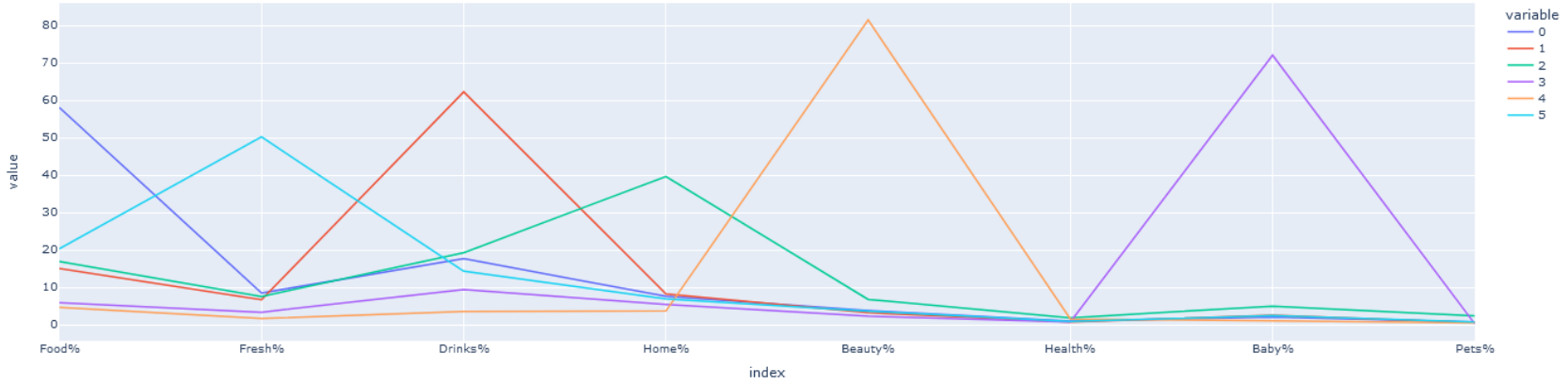
- The optimal number of clusters:

Approach: Road map of the task

4) Clustering Approach: 1) K-means

- It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum.

- The optimal number of clusters: Silhouted method (6)

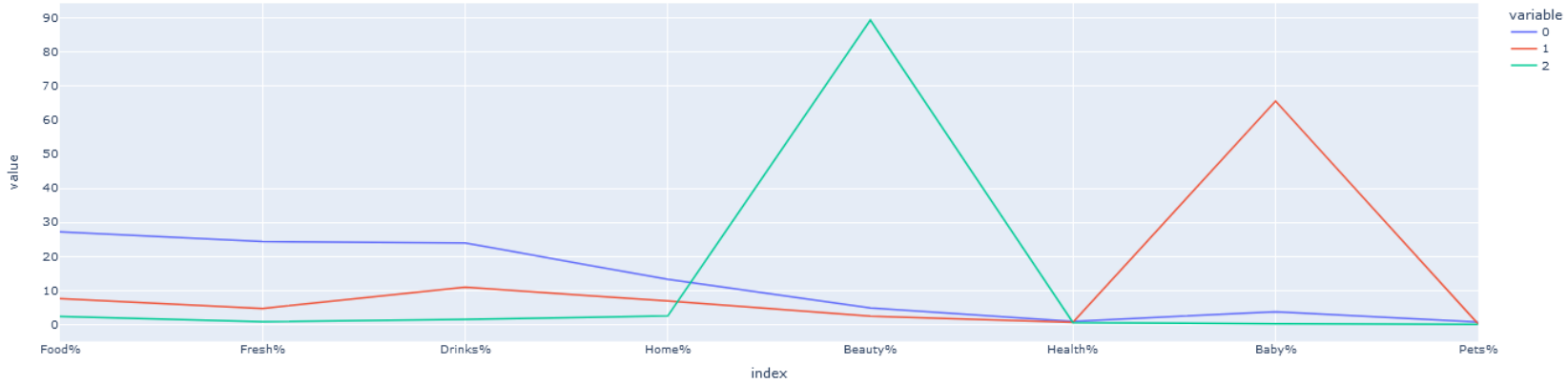


Approach: Road map of the task

4) Clustering Approach: 2) Mean shift

Given a set of data points, the algorithm iteratively assigns each data point towards the closest cluster centroid and direction to the closest cluster centroid is determined by where most of the points nearby are at.

- The optimal number of clusters: 3



Approach: Road map of the task

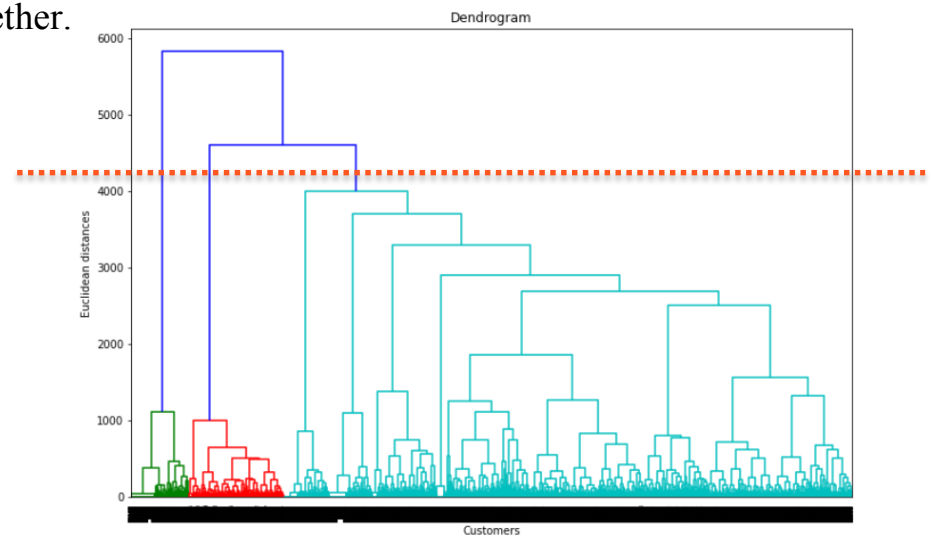
4) Clustering Approach: 3) Agglomerative Hierarchical method

- A **Hierarchical clustering** method works via grouping data into a tree of clusters. Hierarchical clustering begins by treating every data points as a separate cluster. Then, it repeatedly executes the subsequent steps:

  Identify the 2 clusters which can be closest together

  Merge the 2 maximum comparable clusters. We need to continue these steps until all the clusters are merged together.

- The optimal number of clusters: 3



Approach: Road map of the task

4) Clustering Approach: 4) Gaussian Mixture Models (GMM)

Gaussian Mixture Models (GMMs) assume that there are a certain number of Gaussian distributions, and each of these distributions represent a cluster. Hence, a Gaussian Mixture Model tends to group the data points belonging to a single distribution together.

- The optimal number of clusters: 2

Results

Which one has the best performance?

Method of validation of consistency: 1) silhouette method
2) Davies Bouldin

Algorithm	Davies Bouldin	Silhouette Score	Number of Clusters
K-Means	0.9955	0.3495	6
Mean Shift	0.887	0.303	4
GMM	0.5009	0.62608	2
Agglomorative Hierarchical	0.966	0.2059	3

TEAM 3

8:54-9:06

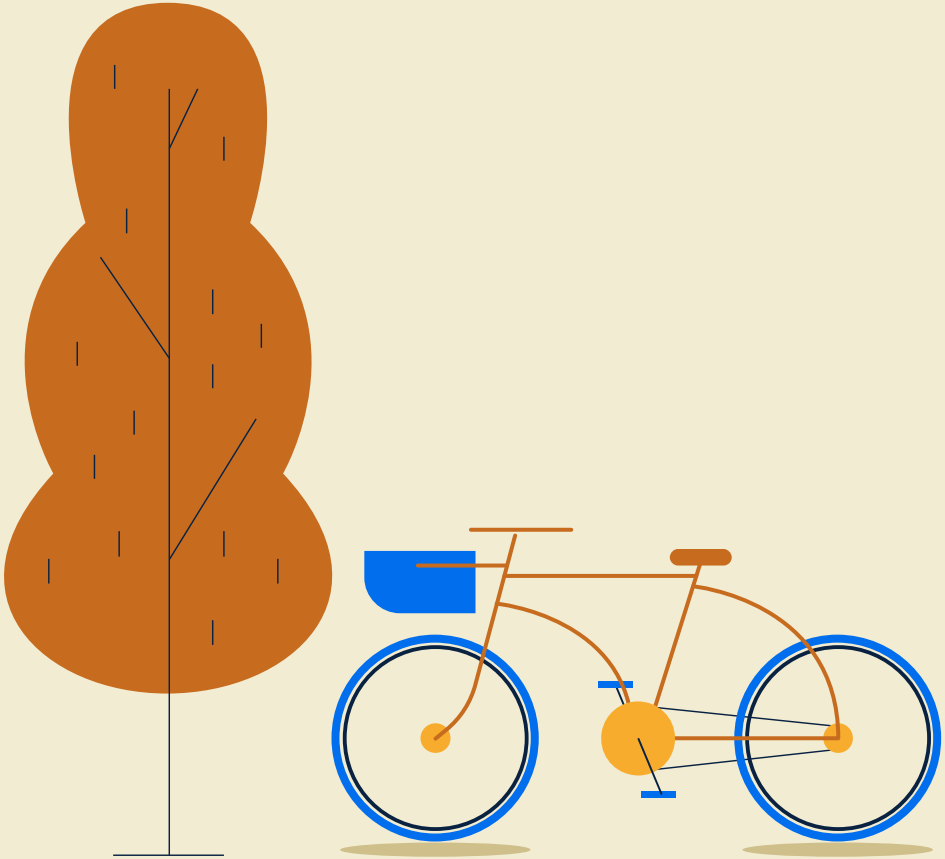
Demand Prediction

with 

Jitsama Tanlamai 11299488
Koray Yenal 11302084

HEC Montreal 2021 - Machine Learning for Large-Scale Data Analysis and Decision Making (MATH80629A)

Contents

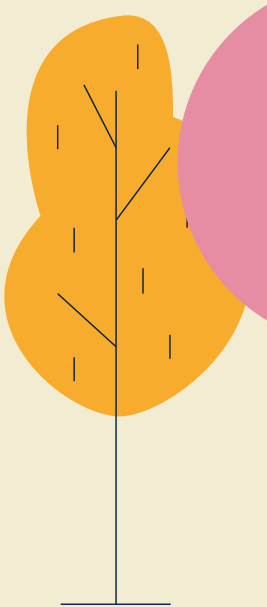


1

Background

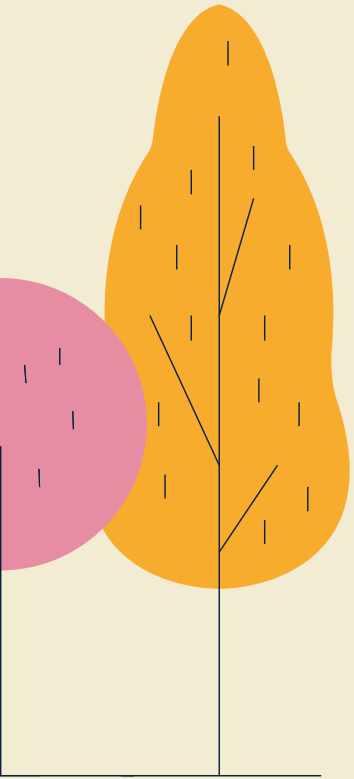
Context

- Bixi is a bike-rental company in Montreal
- Some stations has a shortage in bikes
- The others face an oversupply of bikes.
- Bixi has opportunity cost, lost customers and lost revenue.
- Solution: Accurate prediction of demand across time



Context

- Goal: accurately predict the bike-sharing usage of Bixi
- How: Cluster stations, and compare various machine learning methods
- Test the final model in the real setting



2

Data Preparation and Model Development

Data Retrieval

Ride and Stations

Weather (hourly)

Canada

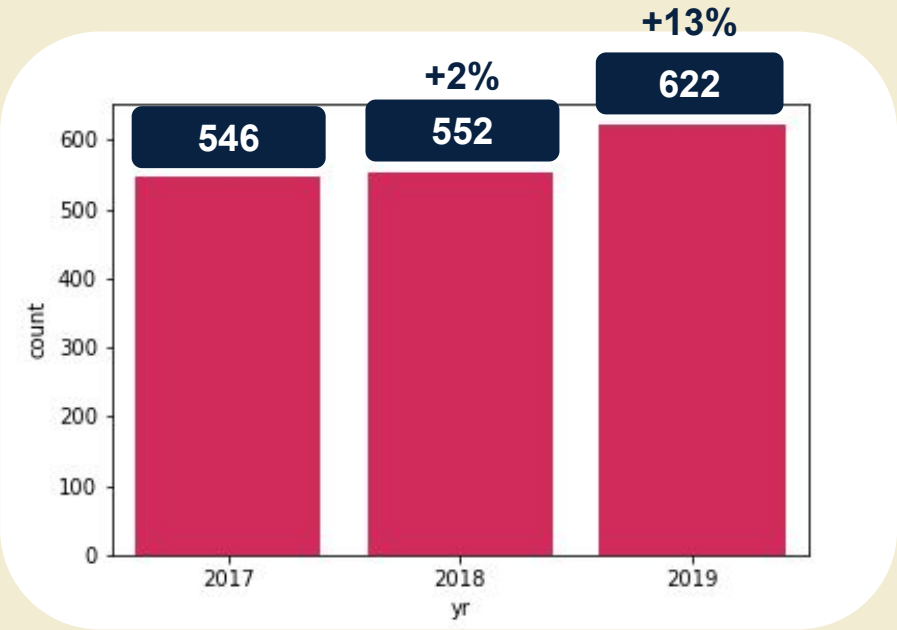
Shortest Route Distance

OpenStreetMap

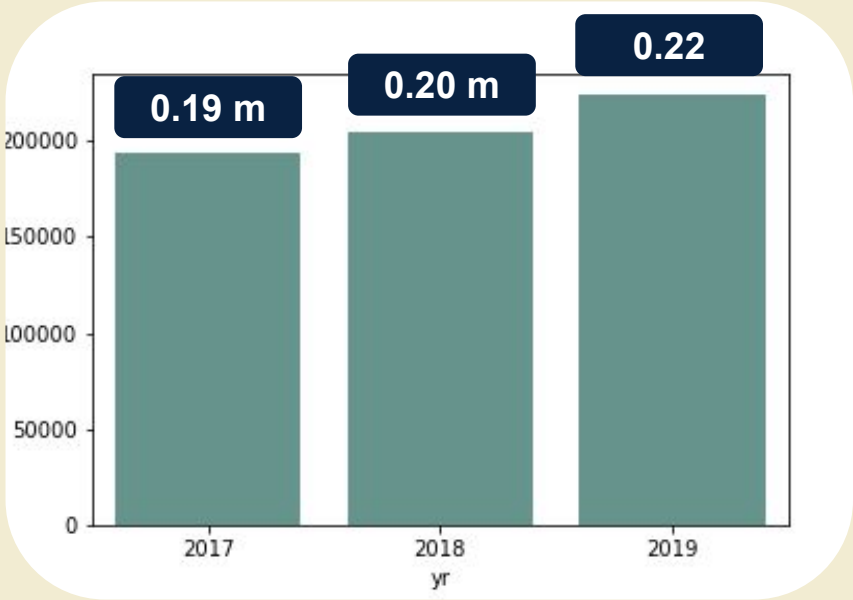
Holiday

Holiday API™

Number of Stations



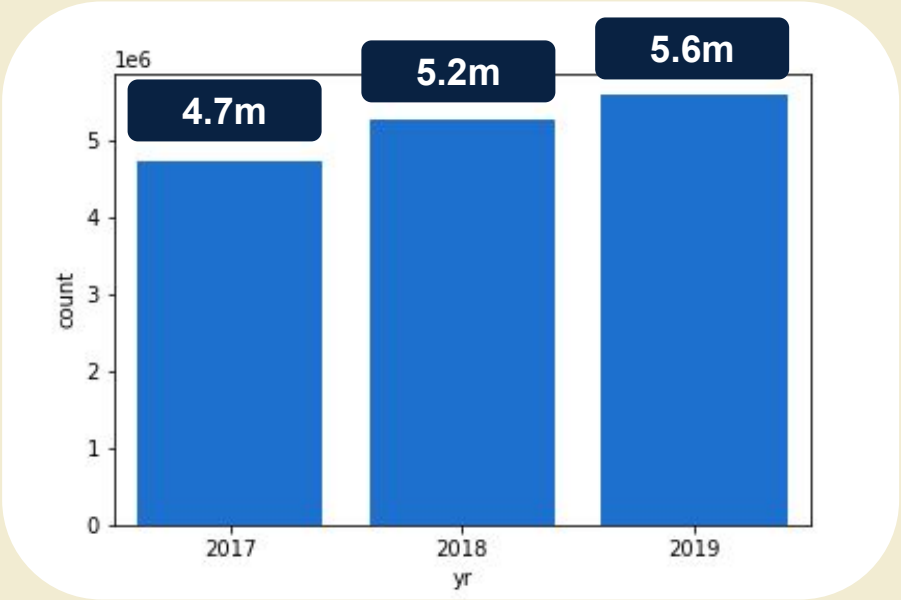
Number of unique routes

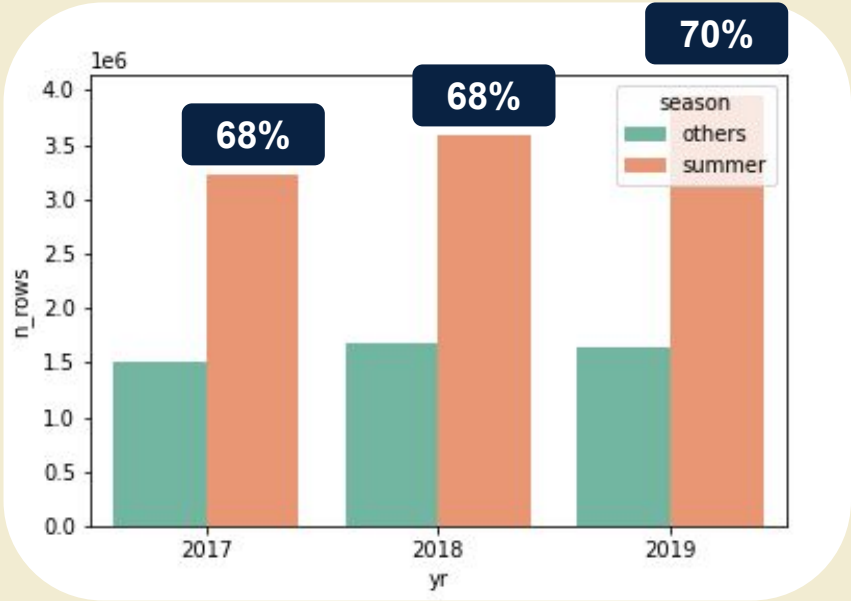


Data Source: https://bixi.com/en/open-data

Assumption: If a new station opens, there must be at least one trip

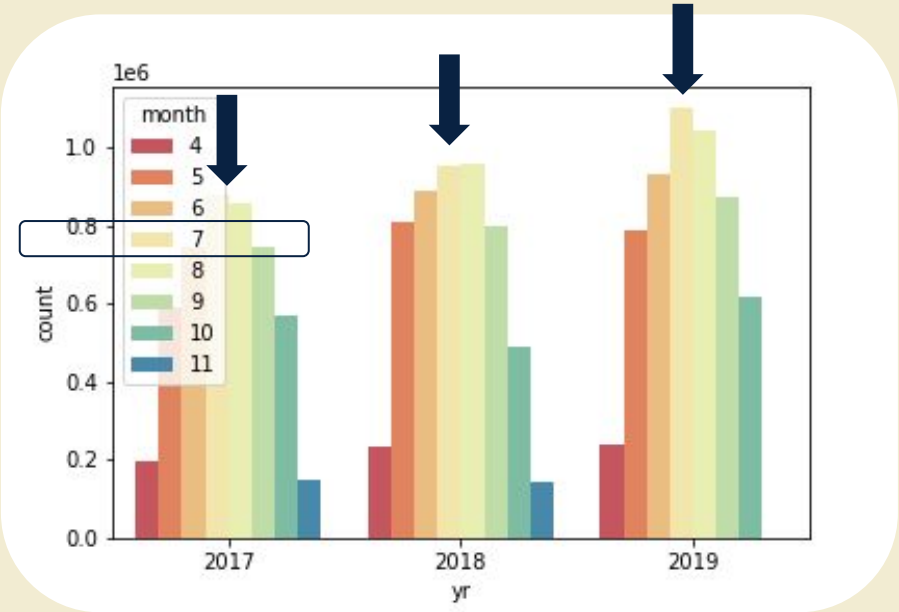
Number of Trips (2017 - 2019)





Data Source: https://bixi.com/en/open-data

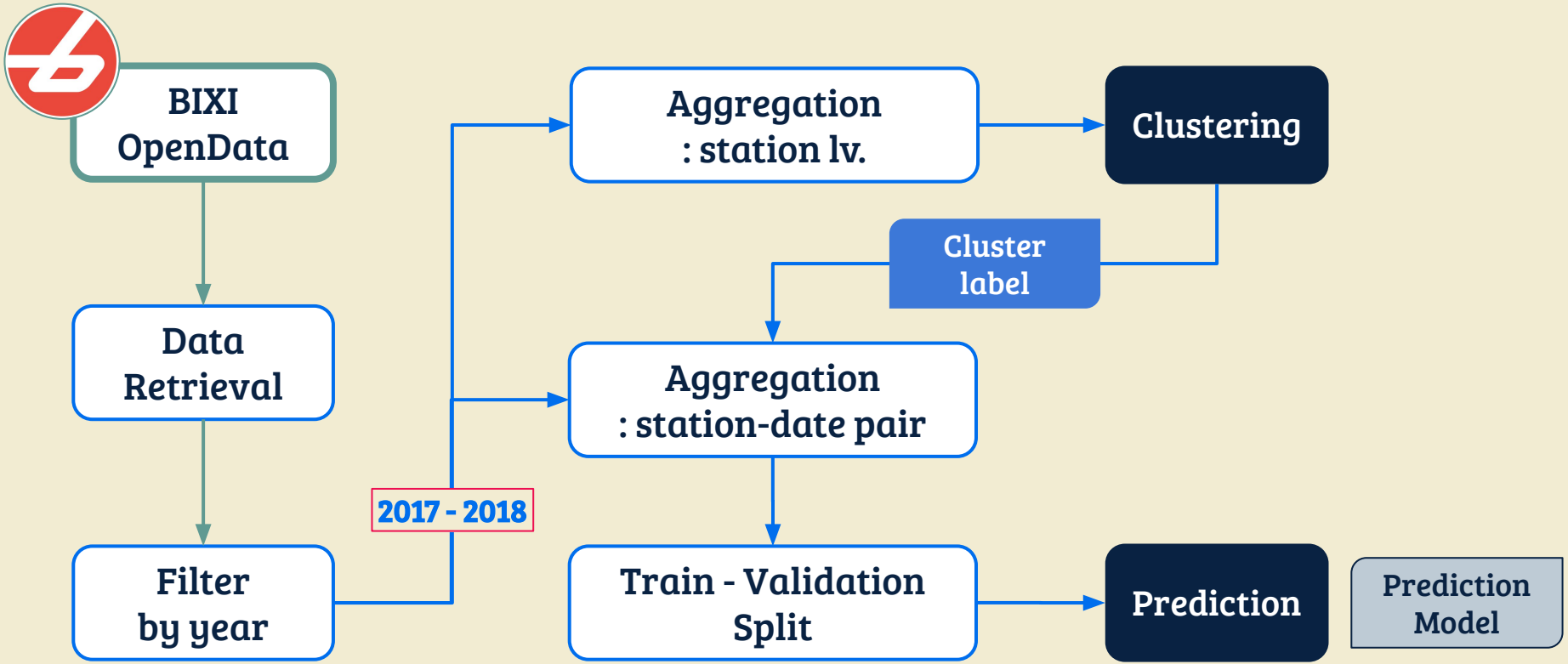
Number of Trips (2017 - 2019)



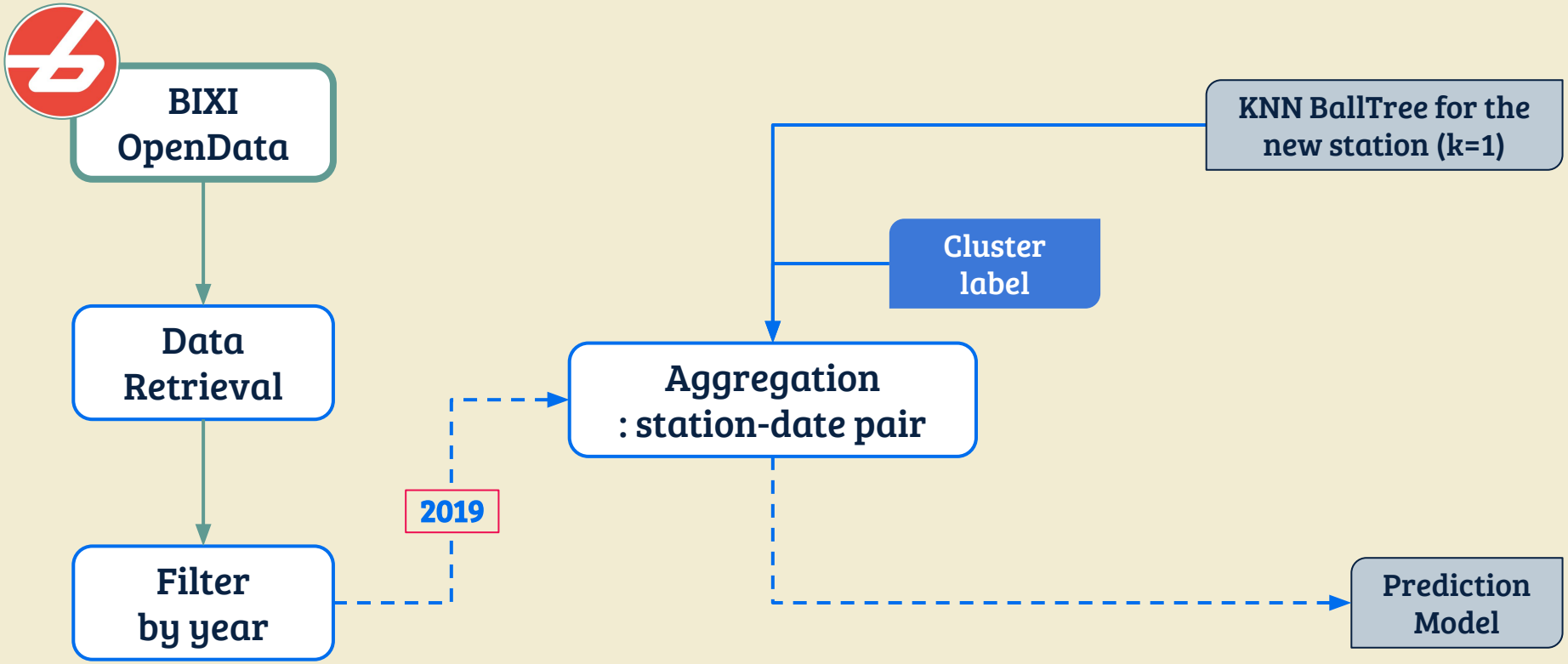
July		
Year	# Trips	of Total
2017	0.88 m	19%
2018	0.95 m	18%
2019	1.1 m	20%

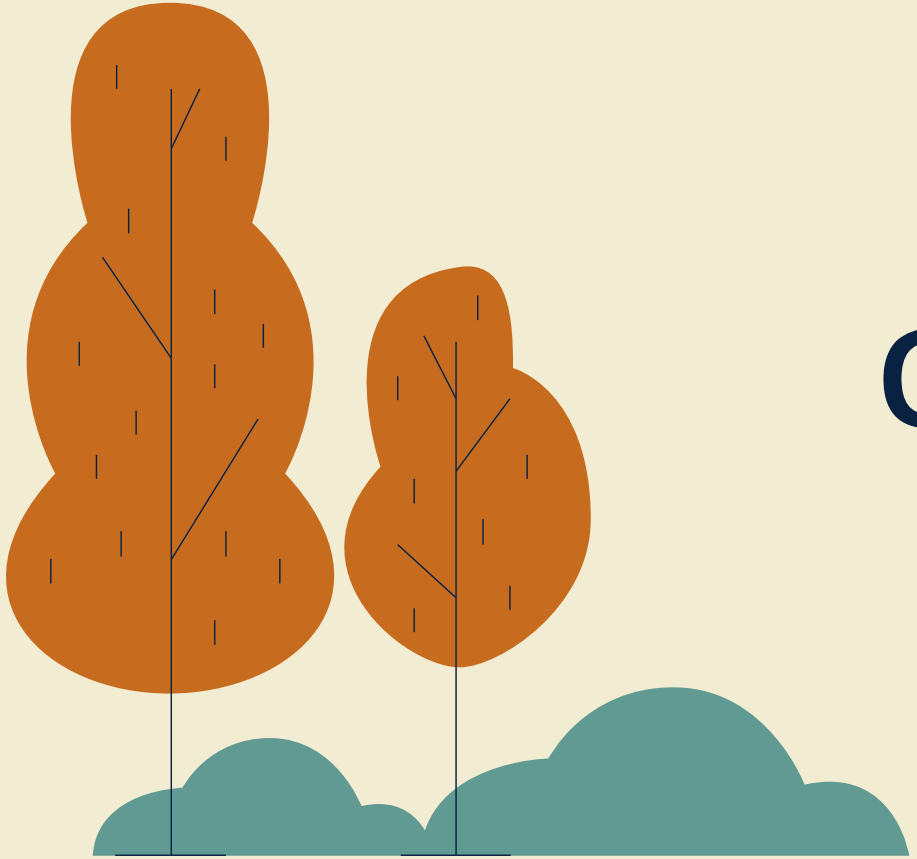
Data Source: https://bixi.com/en/open-data

Workflow - Model Development



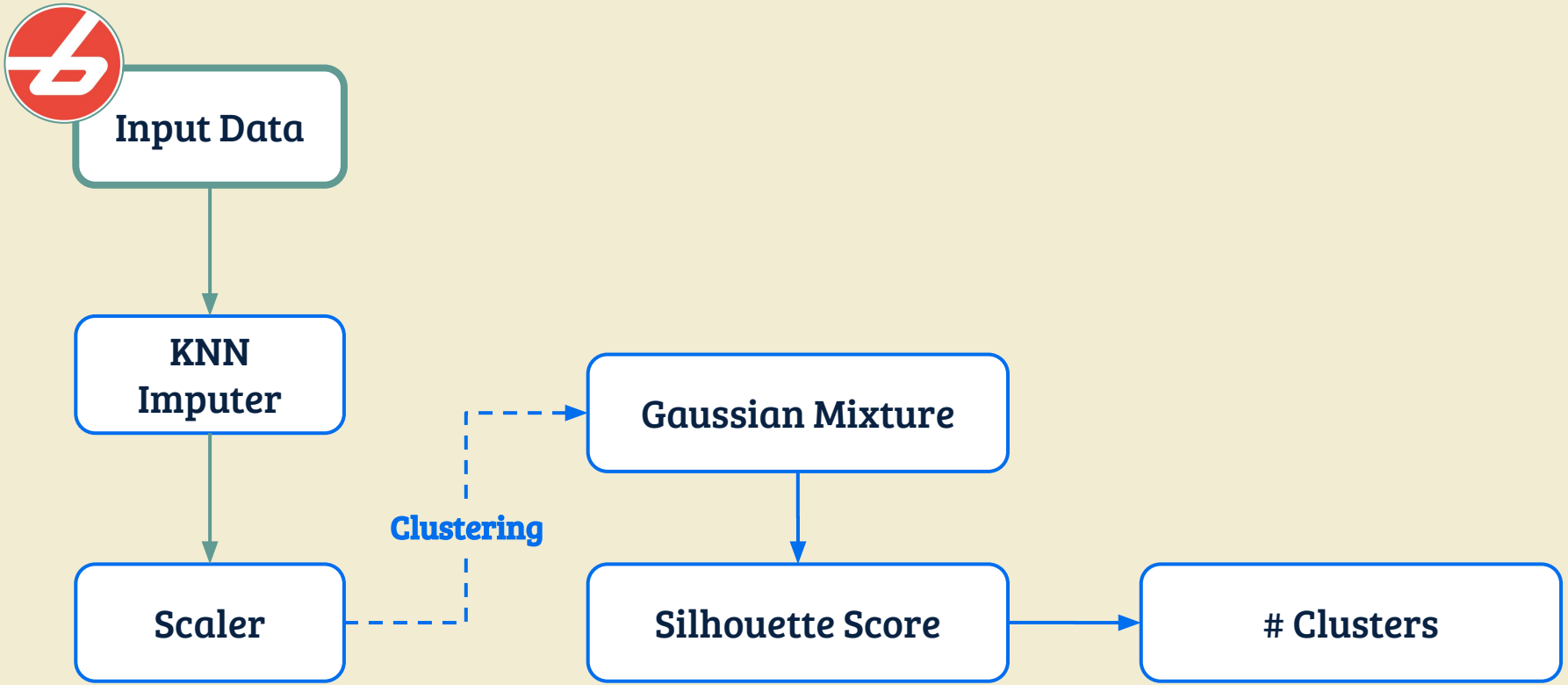
Workflow - Field Experiment





Clustering

Clustering Workflow



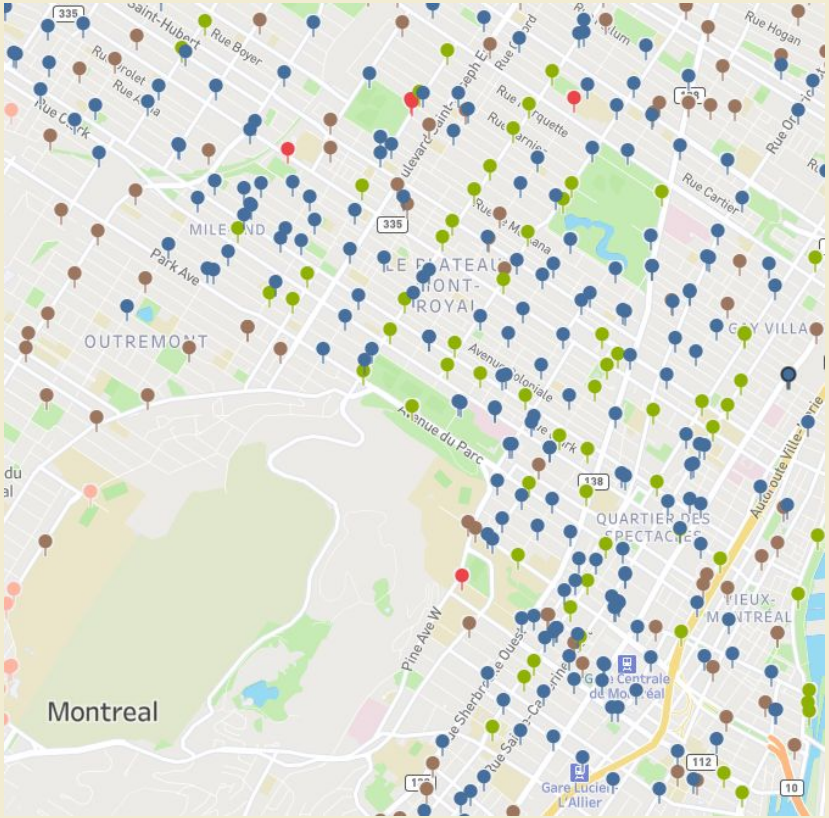
Clustering Features

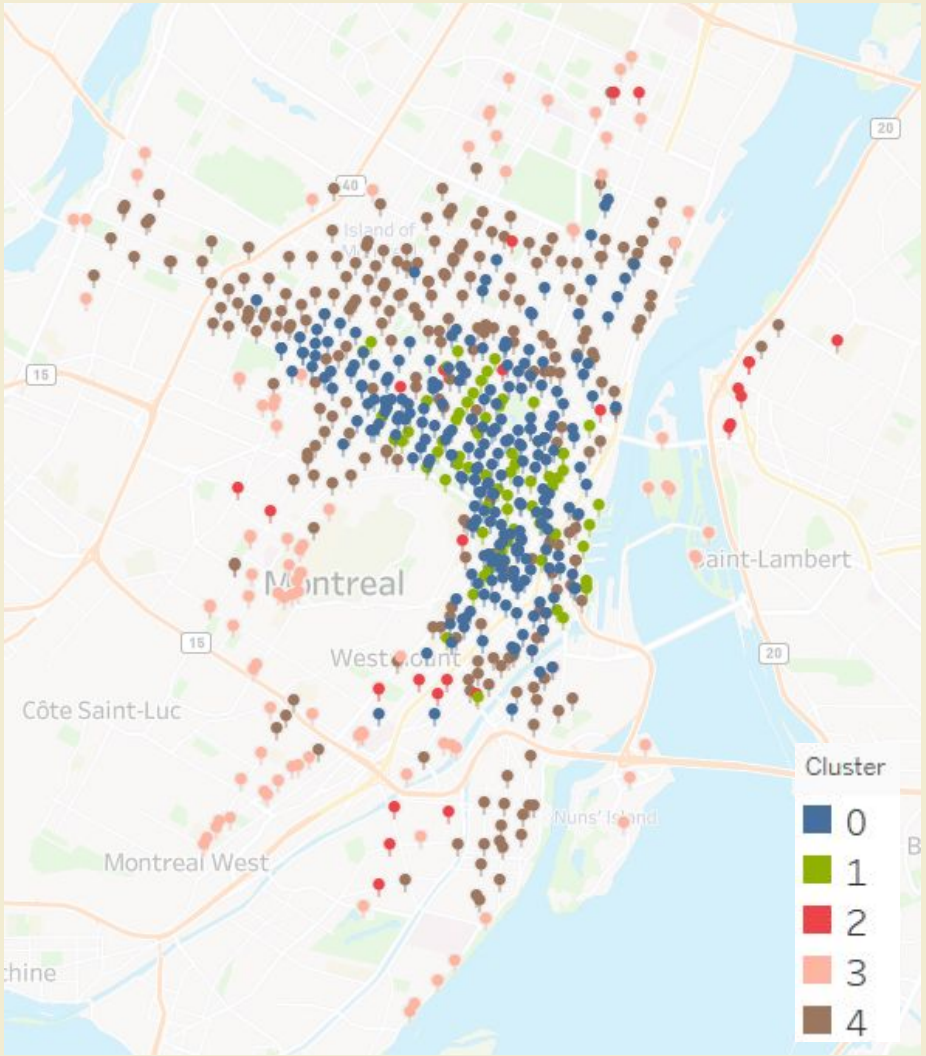
- **Start Station (new code)**
- Coordinate (lat, long)

- Avg number of trips (per day weekday, weekend, holiday, part of the day)
  - Morning: 5 am – 12 pm
  - Afternoon: 1 – 5 pm
  - Evening: 6 – 9 pm
  - Night: 10 pm – 4 am
- Avg number of unique destination
- Avg number of unique origin station

- Avg bike distance
- Avg difference between walk and bike distance
- Avg trip duration

Result

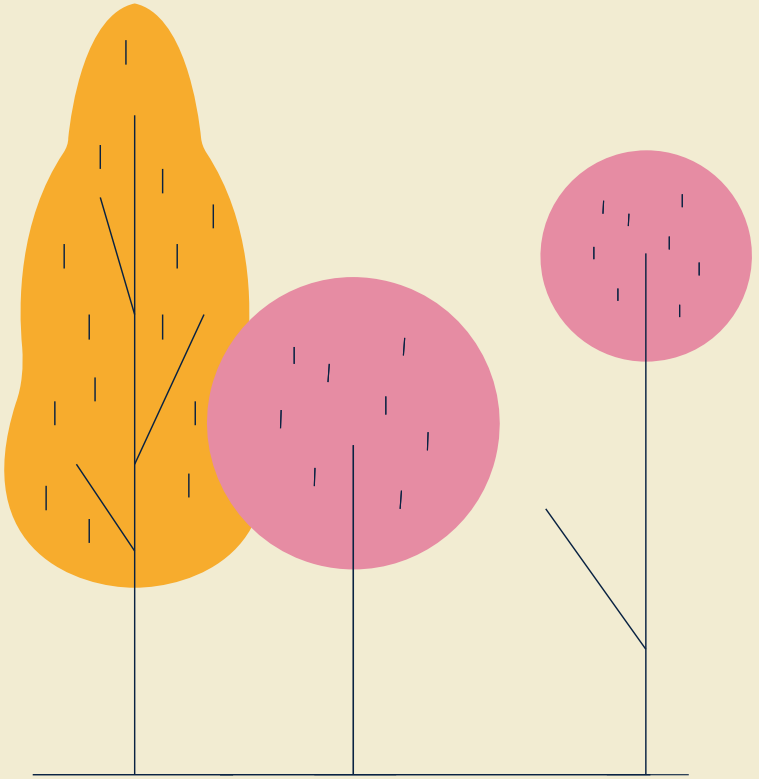

Montreal
OUTREMONT
MILE END
Park Ave
Rue Boyer
Rue Hogan
Rue Cartier
Avenue du Parc
Avenue Bloomfield
Pine Ave W
Rue Sherbrooke Ouest
Gare Centrale de Montréal
Gare Lucien-L'Allier
335
138
112
10


Montreal
West Island
Côte Saint-Luc
Montreal West
Saint-Lambert
Nuns' Island
Cluster
0
1
2
3
4
40
15
20

Result

Cluster name	Trip Characteristics
Working	High freq, but shortest duration and large decrease in distance during holiday
Afternoon Rush	High freq, but each has short distance and duration, and peak at afternoon
Long Distance	Large diff distance between walk and bike
Suburban	Low freq, but each has long distance and duration
Indifferent	Lowest diff distance between walk and bike

Time-series Prediction



Prediction features

Static features

- **Start station (new code)**
- **Date**
- Is weekend
- Is holiday
- Cluster number

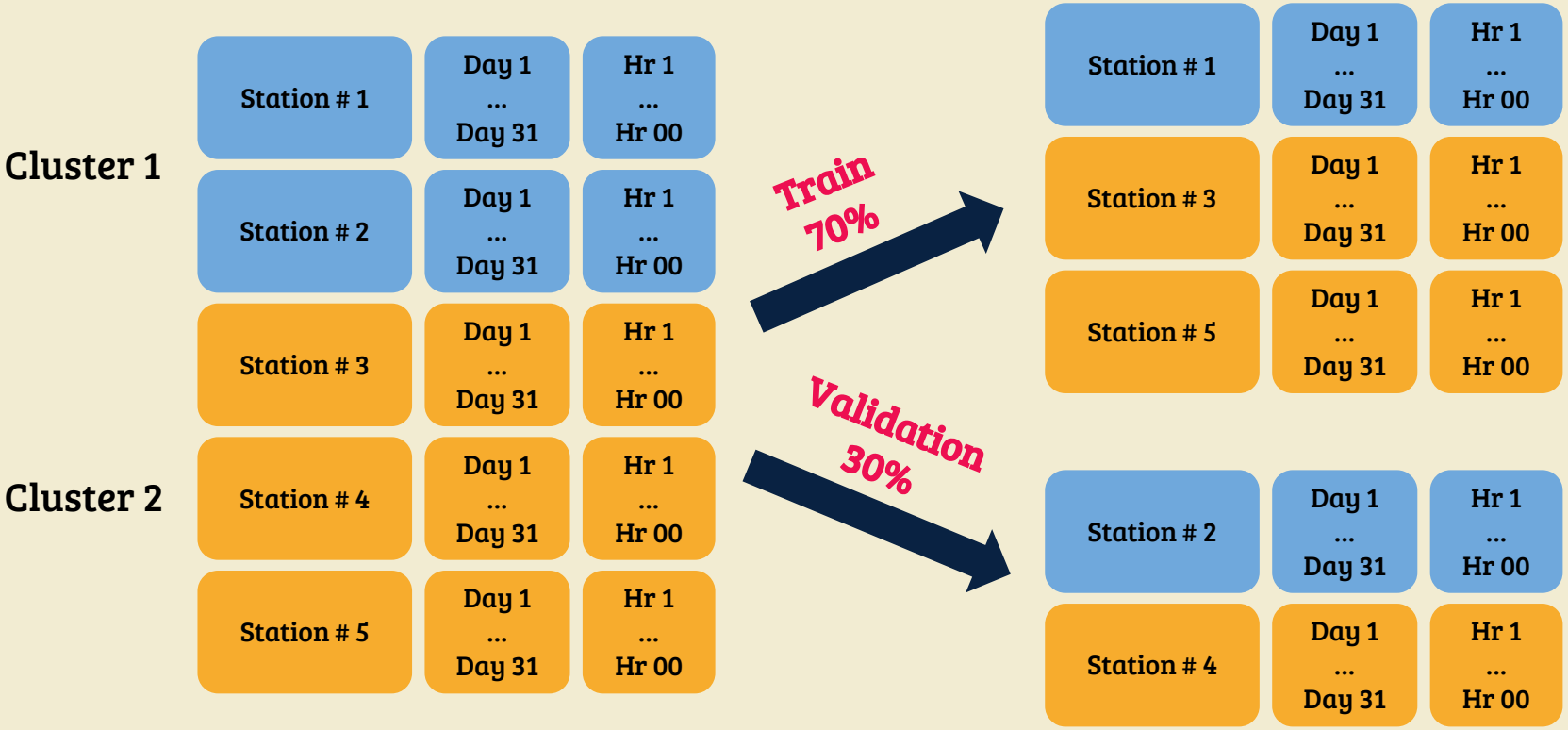
Dynamic features across hour

- **Time (round to the closest hour)**
- Temperature
- Humidity
- Wind direction and speed
- Visibility
- Average bike distance
- Average the difference between bike and walk distance
- Average trip duration

Target

No of trips in the **next** hour

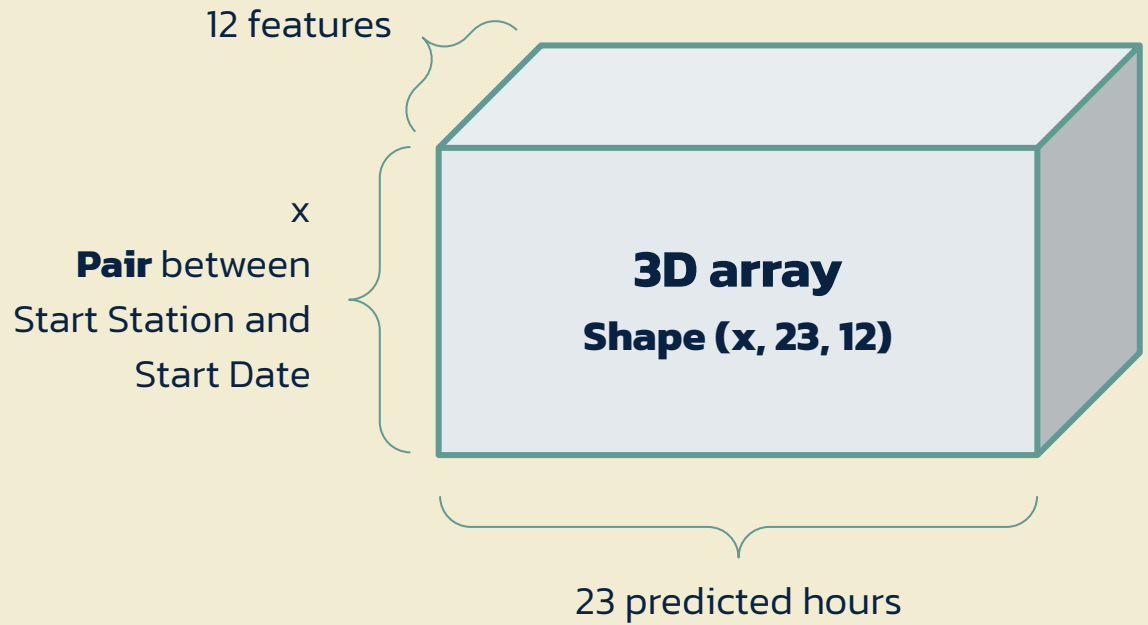
Train-Validation Split



Data Structure

Data

- X train shape = (26494, 23, 12)
- X valid shape = (6731, 23, 12) ~ 20%

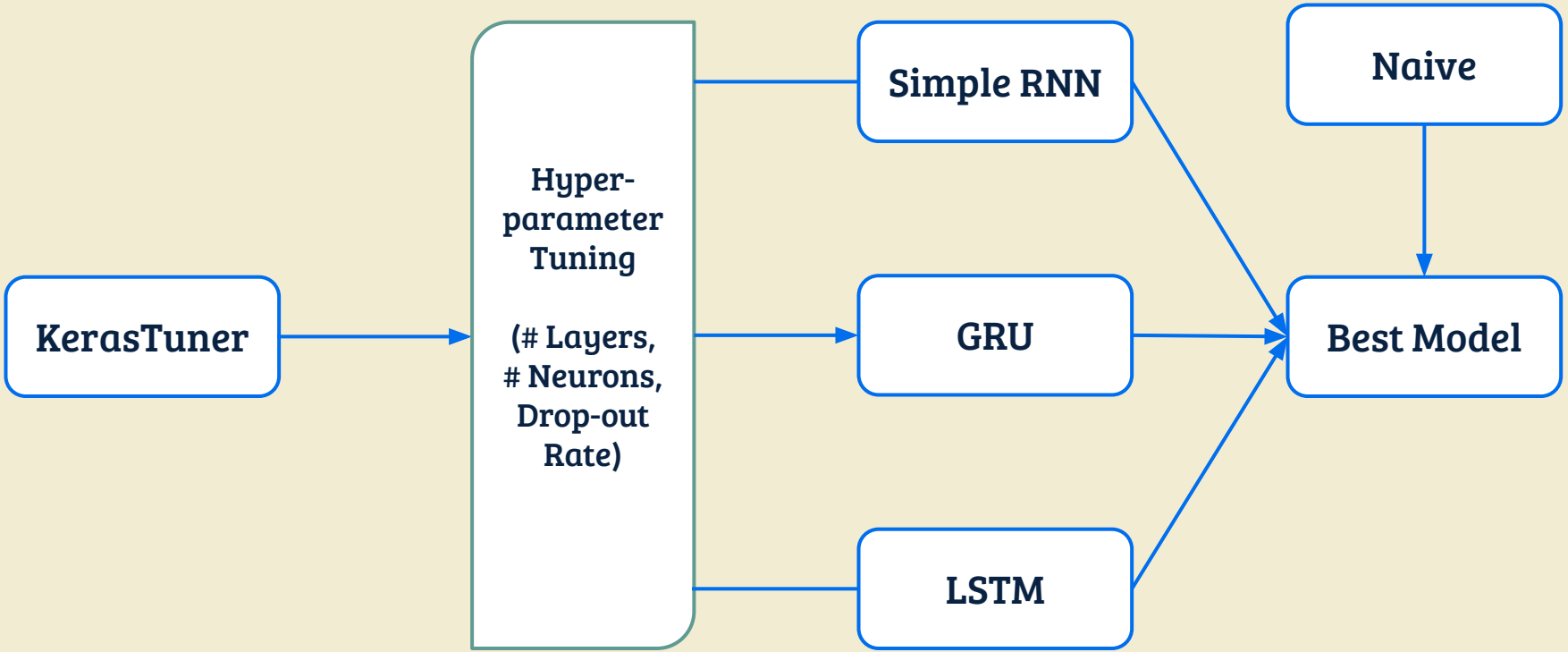


Prediction

Algorithms

- Naive (Baseline)
- Simple RNN
- LSTM
- GRU

Hyperparameter Tuning



Performances
Mean Squared Error (MSE)

Model	Train	Valid
Naive	8.9	9.8
RNN	7.4	8.5
GRU	6.6	7.6
LSTM	6.1	7.4

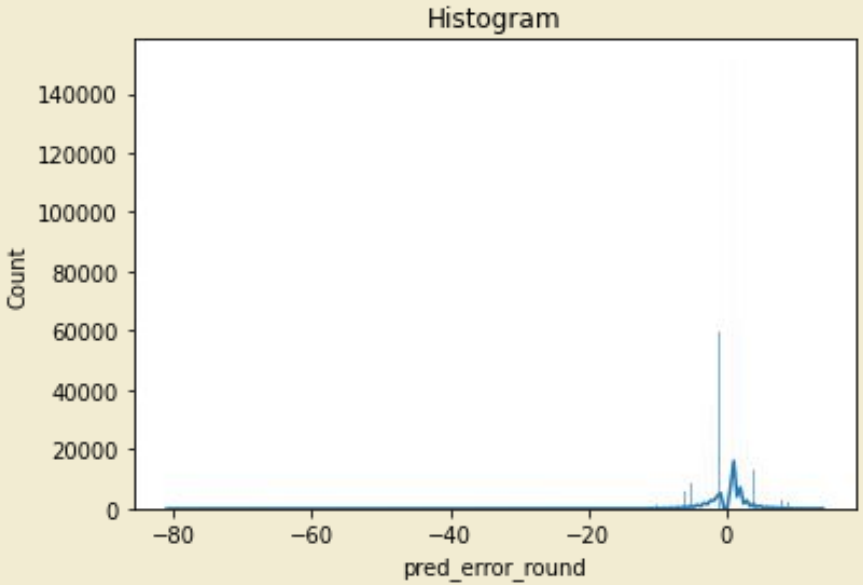
3

Testing

Result

X test shape = (18.935, 23, 12)

Model : Stacked LSTM with (Dropout = 0.1, NN = 96)



	Error Pred - Actual = # trips*
Mean	0.2
25 %	-1
50%	1
75%	2

* ceiling if over estimation, floor if under estimation

Analysis of Error (# trips)

New station does not have close old stations (> 1 km)

cust	
N/A	357%
0	24%
1	57%
2	21%
3	13%
4	21%

	Is Weekend	
Hr	No	Yes
1	0%	4%
2	38%	31%
3	47%	33%
4	58%	48%
5	70%	62%
6	81%	74%
7	66%	83%
8	22%	57%
9	32%	2%
10	36%	15%
11	15%	7%
12	10%	5%
13	27%	17%
14	30%	28%
15	25%	26%
16	20%	25%
17	40%	33%
18	55%	47%
19	54%	50%
20	36%	29%
21	29%	26%
22	18%	21%
23	11%	21%

Note: cut-off threshold at 50% (red)

Thank you!

Questions?

TEAM 4

9:06-9:18

HEC MONTRÉAL

Ensemble learning

Hao Sun
Jiacheng Wang

Task

This project aimed at the case of customers default payments. To predict whether customers will repay their loans on time based on their socio-demographic information and history payment information. which is a supervised learning.

Dataset

	information
Attribute Chraacteristics	Integer, Real
Number of Instances	30,000
Number of Attributes	24
Number of Missing Value	0

HEC MONTRÉAL

Dataset

variable	information
LIMIT_BAL	Amount of the given credit (NT dollar).
SEX	Gender (1 = male; 2 = female).
EDUCATION	1 = graduate school; 2 = university; 3 = high school; 4 = others
MARRIAGE	Marital status (1 = married; 2 = single; 3 = others).
AGE	age
PAY_i	the repayment status in ith month
BILL_i	amount of bill statement in ith month
PAY_AMT_i	amount paid in ith month
default payment	default payment (Yes = 1, No = 0),

HEC MONTRÉAL

Approach

Ensemble learning

	Component Learner
1	Logit Regression
2	K-Nearest Neighbor
3	Naive Bayes
4	Support Vector Machine
5	Decision Tree
6	Random Forest
7	Neural Network

Approach

Reasons for choosing Ensemble Learning

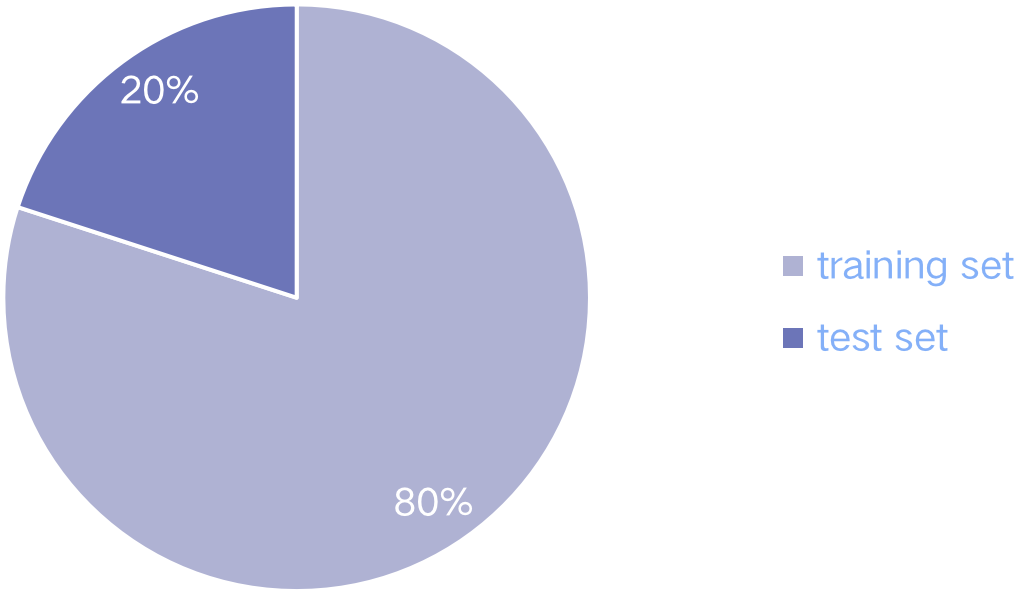
To verify two things:

1.Whether the more complex the model, the better the performance.

2. Whether ensemble learning perform better compared to the single machine learning model

Evaluation Setup

Split data into taining set and test set



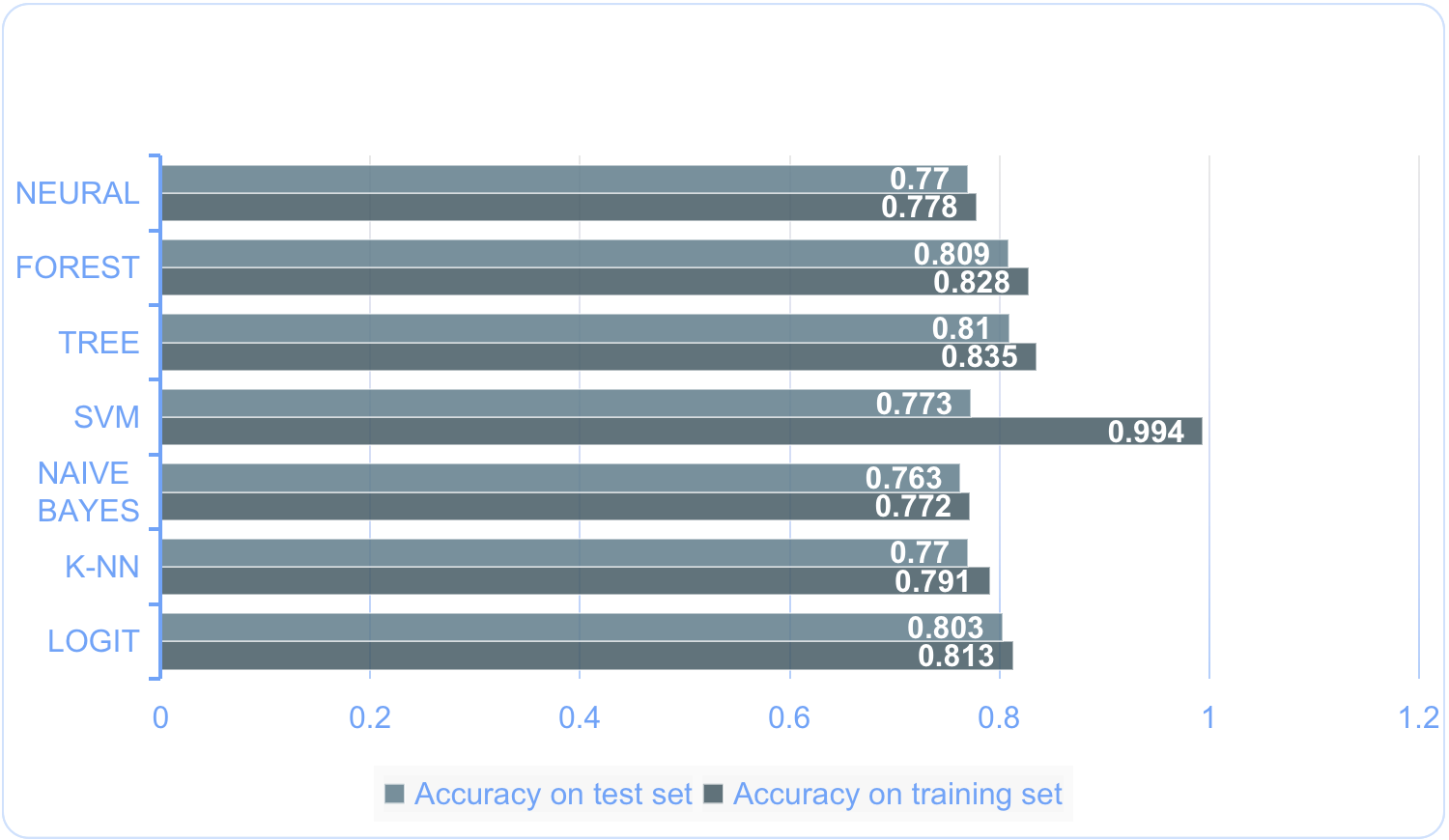
HEC MONTRÉAL

Evaluation Setup

Tuning hyper-parameters: GridsearchCV

Model	Best hyper-parameters
Logit	Penalty: l1 C:1
K-NN	n_neighbors: 14
SVM	C: 1
Tree	max_depth: 4
Forest	max_depth: 6 n_estimators: 44
Neural	activation: identity alpha: 0.001 hidden_layer_sizes: (16, 16, 4) learning_rate_init: 0.001

Results


NEURAL
0.77
0.778
FOREST
0.809
0.828
TREE
0.81
0.835
SVM
0.773
0.994
NAIVE BAYES
0.763
0.772
K-NN
0.77
0.791
LOGIT
0.803
0.813
0
0.2
0.4
0.6
0.8
1
1.2
Accuracy on test set
Accuracy on training set

Results

1. The models with best performance are logit regression, Decision tree and Random forest.

2. Not the more complex the model, the better the performance.

3. There is no best algorithm, only the most suitable one, and this can reflect to some extent the no free lunch theorem

Results

1. The accuracy for ensemble learning : 0.80

2. Compared to models with better performance, the accuracy is very similar.

3. Compared to the model whose performance is not very well, there is an increase in accuracy.

TEAM 5

9:18-9:30

Machine Learning Methods to Predict Electricity Prices

Golshid Aflaki and Nima Rafizadeh

Machine Learning I Course (MATH 80629A)

December 2021

Outline

Introduction

- The prediction of electricity prices is one of the most important routines for an electricity market.
- Accurate prediction of electricity prices is extremely beneficial to market participants in risk management and decision making.
- Broadly speaking, the models used for electricity prices prediction fall into one of two main categories:
  - Statistical methods, e.g., ARMA, ARIMA, GARCH, VAR.
  - Machine learning methods
- Machine learning methods are superior to statistical methods:
  - Because their strength of capturing the nonlinear features and rapid changes.

Introduction

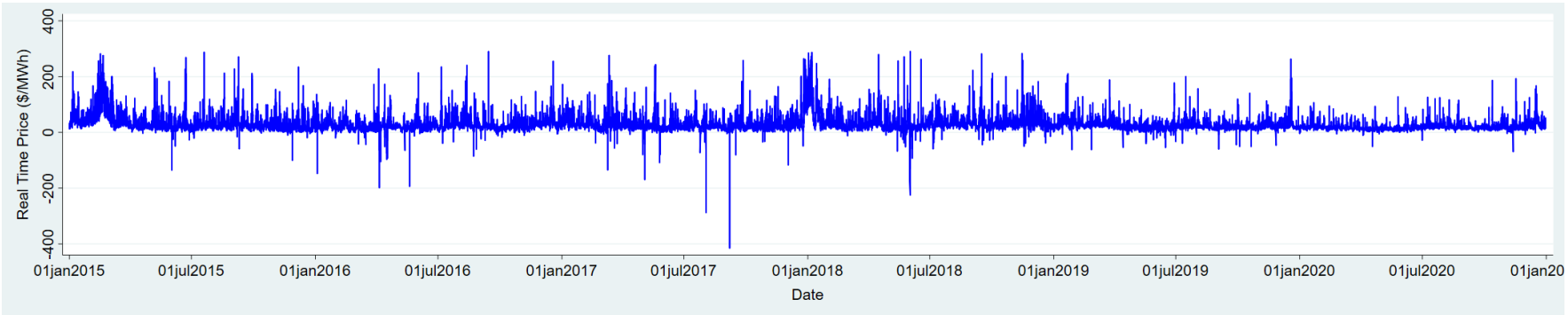
- We use supervised learning models because our data is labeled.
- We aim to address the electricity price prediction issue by:
  - We use a basic model as a benchmark, which considers mean of all electricity prices is a predictor for each new observation.
  - Then a linear regression and regularization methods such as lasso and ridge regressions are applied.
  - Then progressing to more advanced methods such as support vector machine, random forest, neural network, and recurrent neural network.
- We conclude this study by comparing their performance to identify the best model.

Outline

Data

- We exploit the data of the hourly real-time electricity prices in the state of New York from 1th January 2015, 00:00 to 31th December 2020 23:00.
  - Resulting in a total of 52,603 observations.

- Time series of electricity prices:



Data

Table: Descriptive Statistics for the Hourly Electricity Pices

Descriptive Statistics	
Observations	52,603
Mean	27.05
Median	22.07
Maximum	1173.00
Minimum	-415.40
Standard Deviation	26.52
Skewness	9.26
Kurtosis	225.10

Data

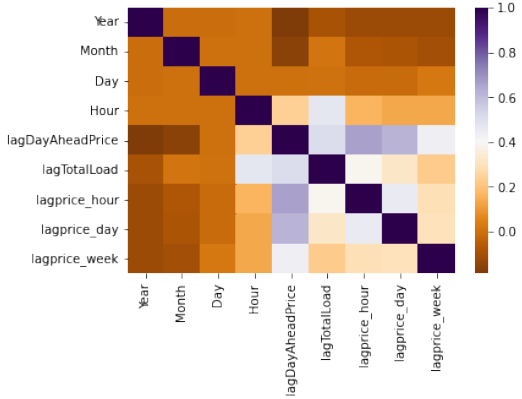
- Head of Data:

	Time	Date	Year	Month	Day	Hour	RealTimePrice	DayAheadPrice	TotalLoad	lagDayAheadPrice	lagTotalLoad	lagprice_hour	lagprice_day	lagprice_week
168	2015-01-08 00:00	2015-01-08	2015	1	8	0	67.723953	75.534004	266.248699	77.315331	240.192700	75.687279	54.872890	26.500668
169	2015-01-08 01:00	2015-01-08	2015	1	8	1	46.695000	70.643997	220.537899	75.534004	266.248699	67.723953	49.495499	32.918835
170	2015-01-08 02:00	2015-01-08	2015	1	8	2	50.571499	67.751999	215.700199	70.643997	220.537899	46.695000	45.589722	29.593000
171	2015-01-08 03:00	2015-01-08	2015	1	8	3	50.260777	66.902664	213.908999	67.751999	215.700199	50.571499	55.981998	22.408890
172	2015-01-08 04:00	2015-01-08	2015	1	8	4	69.597954	73.708664	253.271400	66.902664	213.908999	50.260777	54.897446	21.746778
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
52587	2020-12-31 09:00	2020-12-31	2020	12	31	9	27.504000	22.190666	212.484153	22.235332	205.559729	21.230333	16.688477	12.095612
52588	2020-12-31 10:00	2020-12-31	2020	12	31	10	36.169445	21.195999	217.018450	22.190666	212.484153	27.504000	21.095667	14.020056
52589	2020-12-31 11:00	2020-12-31	2020	12	31	11	47.124290	20.952000	291.815539	21.195999	217.018450	36.169445	15.485778	19.039333
52590	2020-12-31 12:00	2020-12-31	2020	12	31	12	26.322390	20.606001	217.851096	20.952000	291.815539	47.124290	15.570389	23.310278
52591	2020-12-31 13:00	2020-12-31	2020	12	31	13	21.341000	21.253334	217.424115	20.606001	217.851096	26.322390	20.540611	20.464945

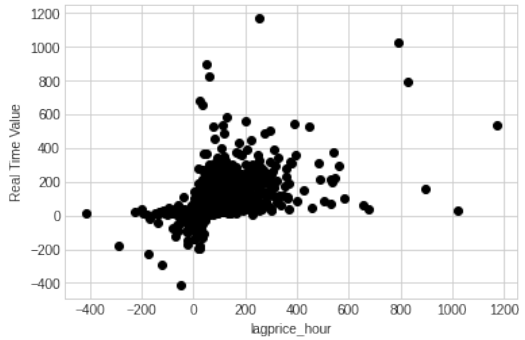
52424 rows × 14 columns

Data

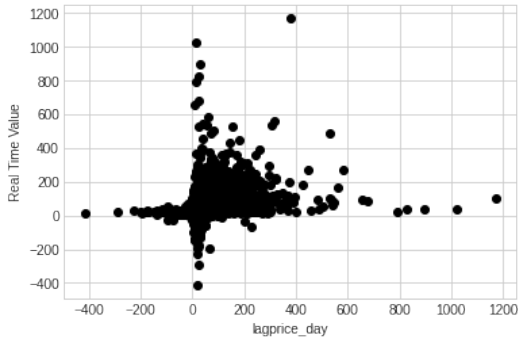
Figure: Data Visualization



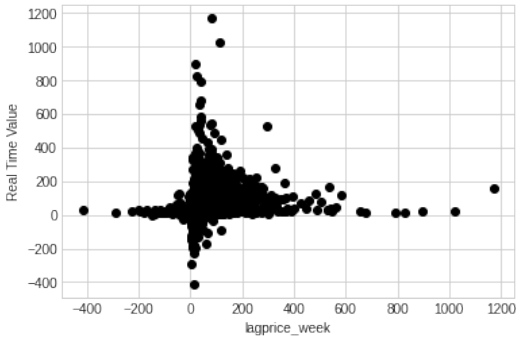
(a) Heat Map plot



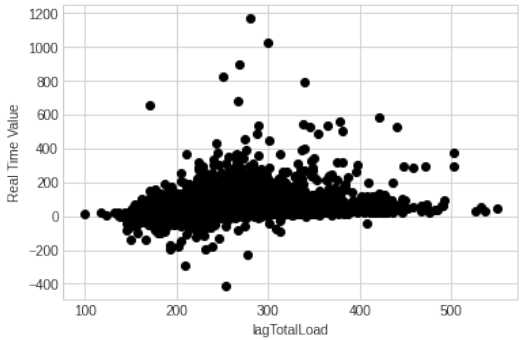
(b) Electricity Price vs. Previous Hour Price



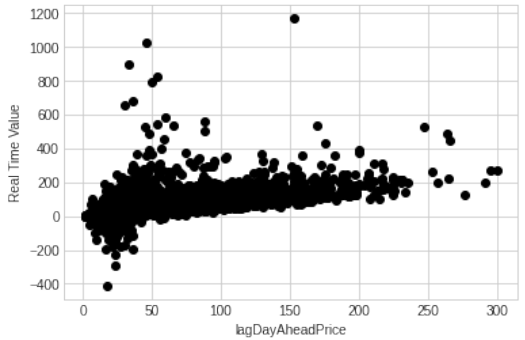
(c) Electricity Price vs. Previous Day Price



(d) Electricity Price vs. Previous Week Price



(e) Electricity Price vs. Previous Hour's Total Load



(f) Electricity Price vs. Previous Hour's Day Ahead Price

Outline

Method and Results

- Real time prices are the target variable in our prediction model.
- The main features are the year, month, day, hour, the previous hour day ahead price, and the previous hour total load.
- We also develop some other features that may improve the precision of the prediction process:
  - Real time prices for the previous hour, day, and week based on the date and real time price columns.

Method and Results

- We begin with linear regression and the methods for regularizing it, namely lasso and ridge regressions.
- We then progress to more advanced methods such as:
  - support vector machine (SVM), random forest (RF), neural network (NN), and recurrent neural network (RNN).
- Our train-test split strategy is based on the K-fold cross-validation for all models except for RNN.
  - Throughout this study, we use $k = 5$.
- We use a basic model as a benchmark, which considers mean of all electricity prices is a predictor for each new observation.
- To compare the accuracy of models, we report the MSE on the averaged across folds.
- To improve the learning ability, we use K-fold cross-validation and grid searching.

Method and Results

Table: Hyperparameters for Each Model

Model	**Hyperparameters**
Linear Regression	-
Lasso Regression	Learning Rate: [1,0.1,0.01,0.001,0.0001,0]
Ridge Regression	Learning Rate: [1,0.1,0.01,0.001,0.0001,0]
SVM	'C': [0.1, 1], 'kernel': ['rbf', 'linear']
RF	'max_depth': [80, 50], 'max_features': [4, 6], 'min_samples_leaf': [20, 15], 'min_samples_split': [50, 100], 'n_estimators': [100, 200]
NN	'hidden_layer_sizes': [(50,50), (20, 30)], 'activation': ['relu','tanh'], 'alpha': [0.0001, 0.05], 'learning_rate': ['constant','adaptive'], 'solver': ['adam']
RNN	-

Method and Results

Table: MSE of the Models

Model	**MSE**
Basic Model	702.679139
Linear Regression	318.842355
Lasso Regression	318.815245
Ridge Regression	318.842354
SVM	322.744595
RF	312.507097
NN	314.651678
RNN	**103.191267**

TEAM 6

9:30-9:42

TEAM 7

9:42-9:54

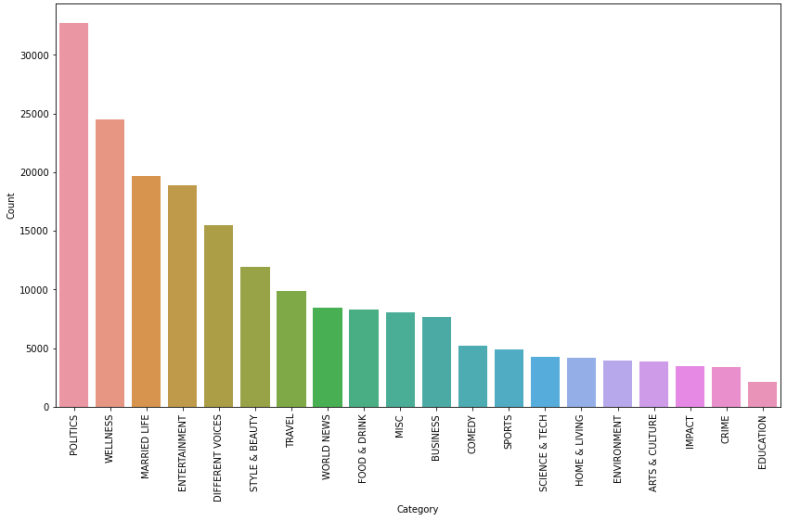
HEC MONTRÉAL

Multiclass Text Classification

—An Application in News Categorization

Michael Gimple
Zichun Liu

Task and Dataset



Task: Assign news articles to specific categories

- Supervised learning
- Classification

Data

- 200,000+ news data from Huffpost from 2012 to 2018
  headline, short description, category, date, author
- Label: 41 categories merged into 20, ex: merged business and money
- Text was cleaned to remove stopwords, punctuation, irregular characters and headline + short description were merged together
- The 10 most common words are: new, one, trump, us, photos, people, time,like, day, get

HEC MONTRÉAL

Approach

Feature representation:

- Tf-Idf -> term frequency and inverse document frequency
- Tokenizer -> encode words to numbers, lower numbers are more important, then transforms text to sequences of numbers, then pad sequences

Learning algorithm:

- Multinomial Naive Bayes -> Multiplies probabilities of text given a category for all text
- Logistic Regression -> sigmoid function, consider dependence of features
- XGBoost -> 'improved' gradient boosting decision tree
- RNN with LSTM -> Using sequential input, 1) encoder, 2)embed -> represent words as vectors, 3) Spatial Dropout to reduce noise and work with reduced features, 4)LSTM, 5)Dense with regularization and softmax, 6)Categorical Cross Entropy loss, 7)Adam/Gradient Descent
- RNN with GRU

Evaluation Setup

- Baseline is majority voting, classify all text as politics category
- Train (and Val) are 2012,2013, 2015-2017, Test is 2014, 2018
- Split is about 80 % to 20% and 64%, 16%, 20%
- Vocabulary size 50,000
- NB: Smoothing Param ranged between 0 - 1 , incremented by 0.1
- XGBoost: max_depth 3, 6, 9; learning_rate 0.05, 0.1, 0.3; min_child_weight 1, 3, 5, 7
- RNN: Learning, Dense L1, L2 regularization, softmax activation, DropoutID, early stopping, 5,082,420 parameters worked best
- Also tried multiple LTSM layers, Bidirectional, ConvID, GRU -> 5,062,620

HEC MONTRÉAL

Results

Feature representation	Learning algorithm	Test Accuracy
Majority Voting	NA	0.15
Tf-Idf	logistic regression	0.54
Tf-Idf	XGBoost(learning rate 0.05, max depth 3, min child weight 1)	0.56
Tf-Idf	Naive Bayes(smoothing = 0.1) Naive Bayes(smoothing = 1)	0.57 0.48
RNN with early stopping	embed=100, spatial dropout1D = 0.2, LSTM(100,dropout, recurrent(0.2)), dense(20), l1,l2=0, learning rate =0.005,	0.62
RNN 2 with early stopping	Same, spatial dropout1D = 0.25, GRU, learning rate= 0.001	0.61

TEAM 8

9:54-10:06

Classifying Trade Direction

Saim Ayberk Sert

HEC Montréal

December 6, 2021

Introduction

- Task:
  - Implementing machine learning methods to offer trade signing model alternatives, if possible, to mostly used Lee and Ready (1991) algorithm.
  - Can machine learning discover Lee and Ready algorithm using same inputs?
  - Can machine learning improve LR algorithm by adding its prediction as an explanatory variable?
- Motivation:
  - Essential to comprehend market dynamics and security price behavior, i.e. market quality, price impact, order dynamics, dealer behavior, spreads.
  - Generally not recorded in intra-day databases, inducing less accurate inferences.
  - Increasing accessibility to intra-day databases provide opportunity to improve trade classification algorithms.

Lee and Ready (1991)

- I evaluate Lee and Ready (1991)'s algorithm to examine its performance as a simple benchmark.

Lee and Ready (1991)'s algorithm

- A transaction is classified as buyer-initiated (seller-initiated) if its execution price is closer to the prevailing ask (bid) quote than to the bid (ask).
- Any transaction effectuated exactly at the mid-quote is classified as buyer-initiated (seller-initiated) if there is an uptick (downtick).
- If the book is empty or there is no price change, zero-upticks are buys, and zero-downticks are sells.

Raw Data

Table 1: A Snapshot from the Transaction Records

TradeNo	Time	BuySell	OrderNo	Price	Quantity	CpiCoeff	Status
784	11:16:58	B	14360	93.075	500000	1	M
784	11:16:58	S	14061	93.075	500000	1	M
785	11:16:58	B	14360	93.1	1500000	1	M
785	11:16:58	S	14028	93.1	1500000	1	M
4383	13:42:44	B	14342	93.0	100000	1	M
4383	13:42:44	S	19469	93.0	100000	1	M

Table 2: A Snapshot from the Order Records

OrderNo	OrderTime	BuySell	Price	Quantity	Balance	AmendTime	LinkedOrderNo	Status
14300	11:13:19.726	S	93.325	5000000	5000000	11:55:15.860	0	W
14334	11:14:47.576	B	93.0	2000000	2000000	11:16:58.788	0	A
14342	11:15:11.627	B	93.0	2000000	1900000	13:47:39.949	0	W
14360	11:16:58.788	B	93.1	2000000	0	11:16:58.788	14334	M

Statistics of dataset

- Buyer- and seller-initiated trades are balanced across train and test set.
- On average, market conditions are similar; however, variation in spread is higher in train set.
- Intervals of traded quantity of bonds, and associated trade values are same across train and test set.

	Price	Quantity	Value	Spread (bps)	Buy	Sell
Train Set						
N	150,177	150,177	150,177	150,177	78,004	72,173
Mean	100.44	1,352,782	1,374,902	12.61		
Sd	3.46	103,738.3	131,094.5	6.98		
Max	105.01	1,488,808	1,562,281	22.28		
Min	95.95	1,193,193	1,192,324	3.41		
Test Set						
N	26,475	26,475	26,475	26,475	14,074	12,401
Mean	94.23	1,199,734	1,156,518	16.12		
Sd	0.59	55,872.46	52,014.68	2.34		
Max	94.64	1,243,777	1,194,210	18.07		
Min	93.55	1,136,885	1,097,175	13.53		

Setup

- I employ a supervised learning approach, training on labeled data where trade direction is accurately signed using order history.
- I experiment 3 machine learning models to look for an effective model that classifies trade direction in Turkish bond market.
- Machine learning algorithms for the classification problem:
  - Logistic Regression
  - Random Forest
  - Neural Network

Setup

- Train and test split in accordance with $80\% - 20\%$
  - Train set includes 12 months of data from 2013-01 to 2013-12.
  - Test set contains first 3 months of 2014.
- Time-series cross validation
  - 7 Folds: 5 months train + 1 month validation with sliding window.

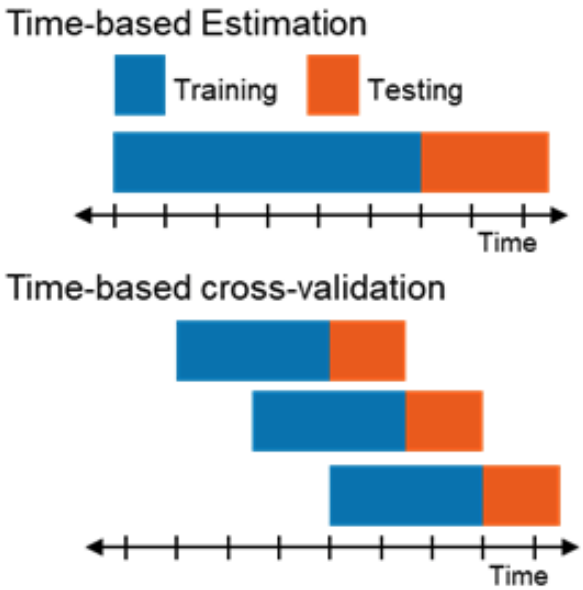


Figure 1: Time based cross validation
(Source: *https://towardsdatascience.com/time-based-cross-validation-d259b13d42b8)*

Hyper-parameter tuning

- Conducted a grid search
  - 54 candidate for each 7 folds, totalling 378 fits.

Model	Hyper-parameter	Grid
Logistic Regression	Penalty	['l1', 'l2']
	C	logspace(-4, 4, 5)
Random Forest	n_estimators	[10, 40, 70 ,100]
	max_features	[6, 9, 12, 15, 18, 21]
Neural Network	hidden_layer_size	[(8,),(8,4,2),(6,3),(12,6)]
	learning_rate_init	[0.1,0.01]
	alpha	[0.001, 0.01,0.1]

Feature Sets

Variable	{1}	{2}	{3}	{4}
Price	✓	✓	✓	✓
Price lag 1	✓	✓	✓	✓
Price lag 2	✓	✓	✓	✓
Price lag 3	✓	✓	✓	✓
Mean price of last 3 trade	✓	✓	✓	✓
Min price of last 3 trade	✓	✓	✓	✓
Max price of last 3 trade	✓	✓	✓	✓
Sd of last 3 trade prices	✓	✓	✓	✓
Trade day	✓	✓	✓	✓
Trade month	✓	✓	✓	✓
Trade year	✓	✓	✓	✓
Trade hour	✓	✓	✓	✓
Trade minute	✓	✓	✓	✓
Trade second	✓	✓	✓	✓
Weekday of trade	✓	✓	✓	✓
Best Ask Price	✓	✓	✓	✓
Best Bid Price	✓	✓	✓	✓
Mid Quote	✓	✓	✓	✓
Quantity	✓	✓	✓	✓
Volume	✓	✓	✓	✓
Volume lag 1		✓		✓
Volume lag 2		✓		✓
Volume lag 3		✓		✓
Mean volume of last 3 trade		✓		✓
Sd of last 3 trade volumes		✓		✓
Lee and Ready flag			✓	✓

Results

- Best fitted model is Logistic regression with L1 filter in feature set 3 and 4, inducing the model yield a sparse model concentrated around LR prediction variable.

	Parameters	Test Score
Best Fits		
{1} - Random Forest	(max_features=6, n_estimators=100)	0.635
{2} - Logistic Regression	(penalty='l1', C=1)	0.581
{3} - Logistic Regression	(penalty='l1', C=0.01)	0.853
{4} - Logistic Regression	(penalty='l1', C=0.01)	0.853
Benchmarks		
Majority Voting	Vote=1 (Buy)	0.532
Lee and Ready Algorithm		0.849

Conclusion

- Machine learning experiments that use same price inputs with LR method, i.e. feature set 1 and 2, could not discover LR algorithm.
- Adding LR prediction as an input significantly improved the accuracy in test set.
- Best fitted model is Logistic regression with L1 filter in feature set 3 and 4, inducing the model yield a sparse model concentrated around LR prediction variable.
- Further investigation:
  - Expanding time horizon of the data over 20 years.
  - Additional to time series controls, adding controls for cross sectional variation, i.e. across bonds characteristics and market dynamics.
  - Increasing the number of models tested, e.g. boosting models like XGBoost.
  - Incorporating trade direction forecasts into a trade algorithm to control direction of the market & information flow.

THANK YOU!

TEAM 9

10:06-10:18

Prediction for NCAA Basketball Tournament

Tiange Li
tiange.li@mail.mcgill.ca

Task and Dataset

Dataset consists of two sections:

(1) each year's game result: 3 columns [season, winning team, losing team]
data size: from 2002 to 2019 * 64-68 games each year (1175 in total)

(2) each team's general performance and coach data each year (e.g. seed, fg3pct, pt_coach_season_wins)
data size: 40 features* 1175 games

Data transformation:

(1) Build an rnn baseline model based on data section 1 (team-level, supervised regression)

	2002wins	2003wins	....	2017wins	2018wins	2019wins
Team1						
Team2						
Team3.....						

to predict

Task and Dataset

Dataset consists of two sections:

(1) each year's game result: 3 columns [season, winning team, losing team]
data size: from 2002 to 2019 * 64~68 games each year

(2) each team's general performance and coach data each year (e.g. seed, fg3pct, pt_coach_season_wins)
data size: 40 features* 1175 games

Data transformation:

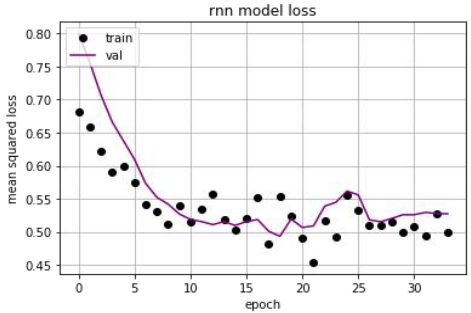
(2) Include both data sections, and build supervised binary classifier at season-game level

				seed1-seed2	fg3pct1-fg3pct2	coachwins1-coachwins2	.....	Result
train	2002	Team1	Team2					1(win)
		Team1	Team3					0(lose)
	.....	.....	.....					
	2017	.....	.....					
val →	2018							
test →	2019							

Baseline results

Data section 1 baseline: n=252

Method: LSTM, dropout, early stopping,gridsearchcv



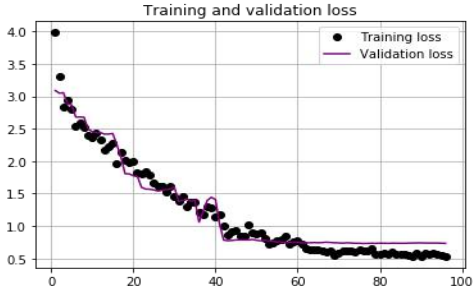
For two teams in a game, the team with a higher RNN-predicted number of wins will be classified as the winning team

Prediction accuracy for season 2018: 77.61%

Prediction accuracy for season 2019: 77.78%

Data section 1&2 baseline: n=1175

Method: Two dense layers with dropout, early stopping,gridsearchcv

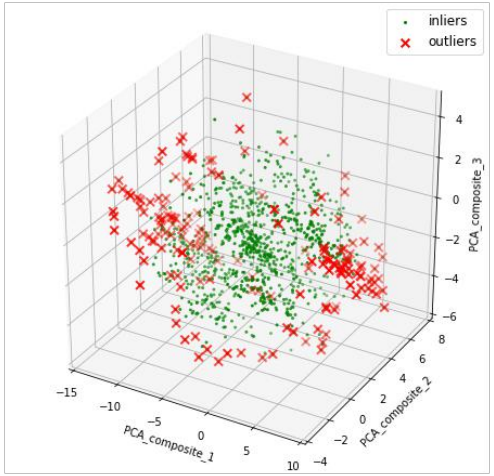


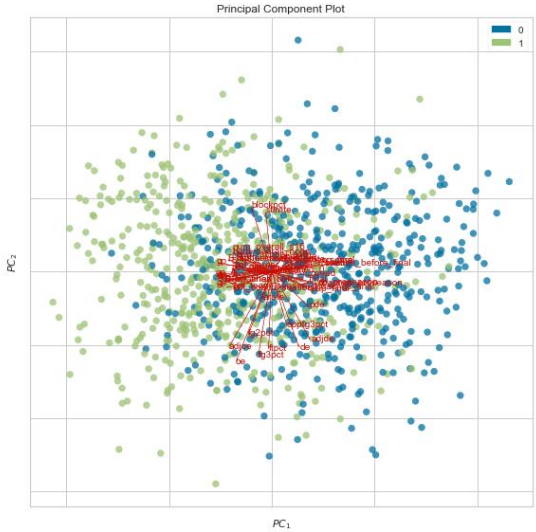
Training accuracy: 72.7%

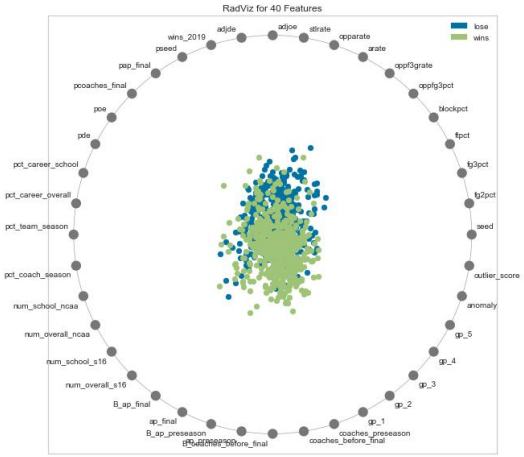
Validation accuracy: 69%

Feature Engineering

1. Anomaly detection: isolation forest
2.Genetic Algorithm: 5 features that can produce strongest offsprings
3. Add Last years performance
4. Remove redundant features

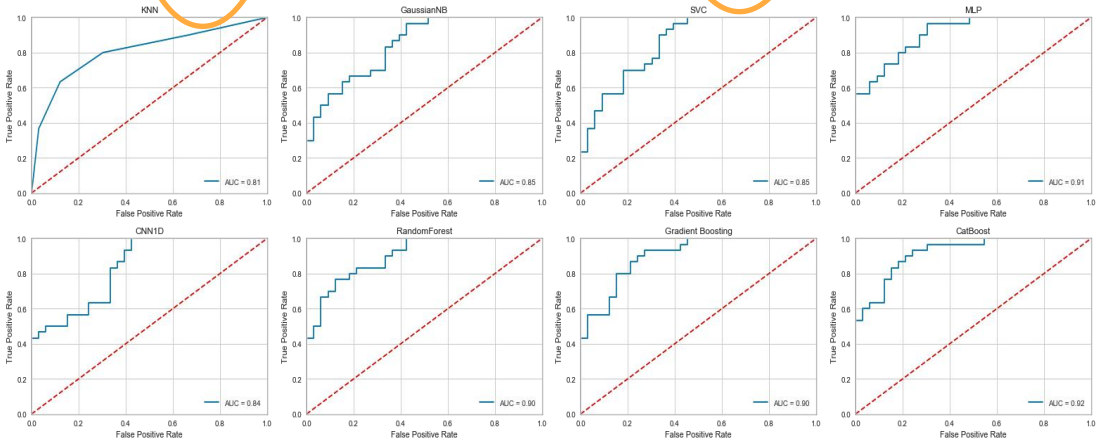






Model Results

Tuned models	baseline MLP	KNN	NB	SVC	MLP	CNN1D	Random Forest	Gradient Boost	CatBoost
Train AUC	72.7%	79.5%	71.8%	72.7%	86.4%	76.3%	94.5%	82.2%	**83.5%**
Val AUC	69%	71.6%	73.1%	71.6%	76.1%	77.6%	79.1%	79.1%	**83.6%**

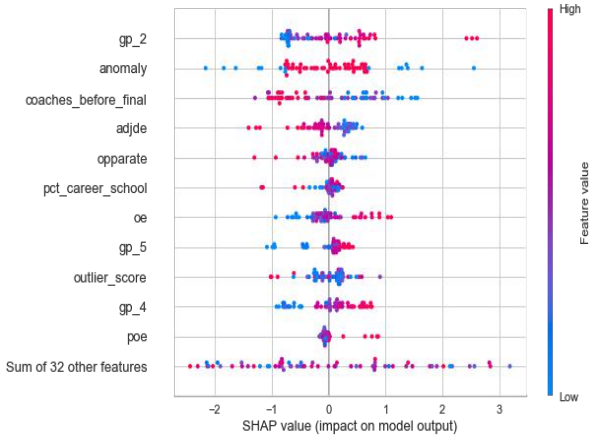


Test accuracy: (based on CatBoost) 83.0%

Result Interpretation

Feature importance plot

(based on SHAP value)



Two examples:

Michigan vs LSU

Kentucky vs Auburn

TEAM 10

10:18-10:30

GROUP HOPE

Birce Adsanver

Motahhareh Safdari Shadlou

Post-disaster Damage Classification for Buildings

Introduction

- Pre-disaster information
  - Features
- Post-disaster information
  - Labels (i.e., damage grade)

Problem Definition

- **Case 1**
  - Building damage information (i.e., labels) is collected randomly.
  - Imbalanced information for labels.
- **Case 2**
  - Building damage information (i.e., labels) is collected based on clustering.
  - Balanced information for labels.
- **Data Availability**
  - 25%
  - 50%
  - 75%

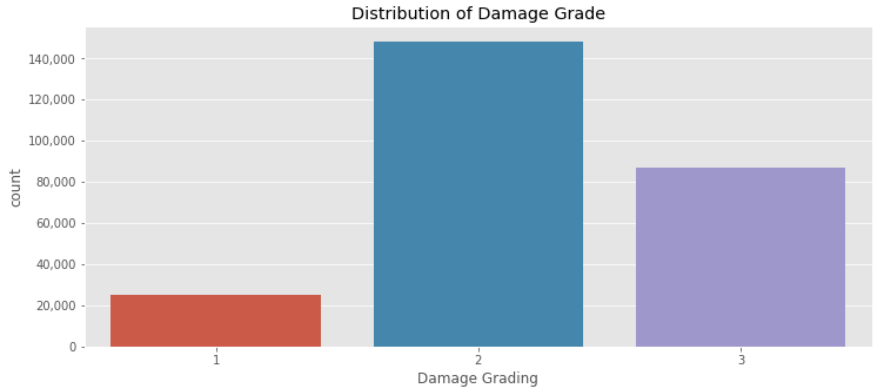
} **Model Training**
Training Set + Validation Set

Data Set

- DrivenData
  - Richter's Predictor: Modeling Earthquake Damage
  - 260601 data points
  - 38 features (Categorical, binary, numerical)

- Labels
  - 1: Low Damage
  - 2: Medium Damage
  - 3: High Damage

Index	count_floors_pre_eq	age	area_percentage	height_percentage
count	260601	260601	260601	260601
mean	2.12972	26.535	8.01805	5.43437
std	0.727665	73.5659	4.39223	1.91842
min	1	0	1	2
25%	2	10	5	4
50%	2	15	7	5
75%	2	30	9	6
max	9	995	100	32



Approach

K Nearest Neighbors	Neural Network
Advantages	
Easy to implement for multi-class problems	Ability to learn complex relationships
No assumption is needed	Works well with large datasets
One hyper-parameter	Fast after training
Intuitive	Flexible
Simple	
Disadvantages	
Curse of dimensionality	Dependency on the training data
Prone to imbalanced data	Expensive Training
Prone to outliers	

Evaluation Setup

- Steps

1. Data Analysis and Data Preparation
3. Creating the Training Set → Case 1: Random; Case 2: Clustering
4. Outlier Detection → IQR
5. Model Implementation and Hyper-Parameter Tuning →
   - Cross-validation (K=5)
   - ❑ K-NN
     - K = [10, 20, ..., 200]
   - ❑ NN
     - Neurons = [2,4,8,16]
     - Alpha = [0.001,0.01,0.1]
     - Learning Rate = [0.1,0.01,0.001]
     - Hidden Layers = [2,4]
6. Model Selection →
   - Baseline: 1-NN
   - Performance Metrics
     - Computation Time (including training time)
     - Accuracy
     - True Positives for Damage Grade 3
7. Feature Selection

Results – Model Comparison

		1-NN			Neural Network				K-NN			
		Test			Test			Train	Test			Train
		Test Accuracy	TP3	Comp. time (s)	Test Accuracy	TP3	Comp. time (s)	Comp. time (s)	Train Accuracy	TP3	Comp. time (s)	Comp. time (s)
Case 1	25%	55%	47%	124	63%	67%	4	40	58%	62%	135	278
	50%	56%	48%	278	63%	69%	0.9	138	59%	63%	147	1048
	75%	57%	50%	148	64%	75%	0.6	227	60%	63%	152	2522
Case 2	25%	55%	47%	125	63%	64%	0.8	50	59%	63%	134	245
	50%	56%	48%	180	63%	66%	1.1	125	60%	63%	182	2350
	75%	57%	50%	152	67%	69%	3	192	60%	65%	340	2678

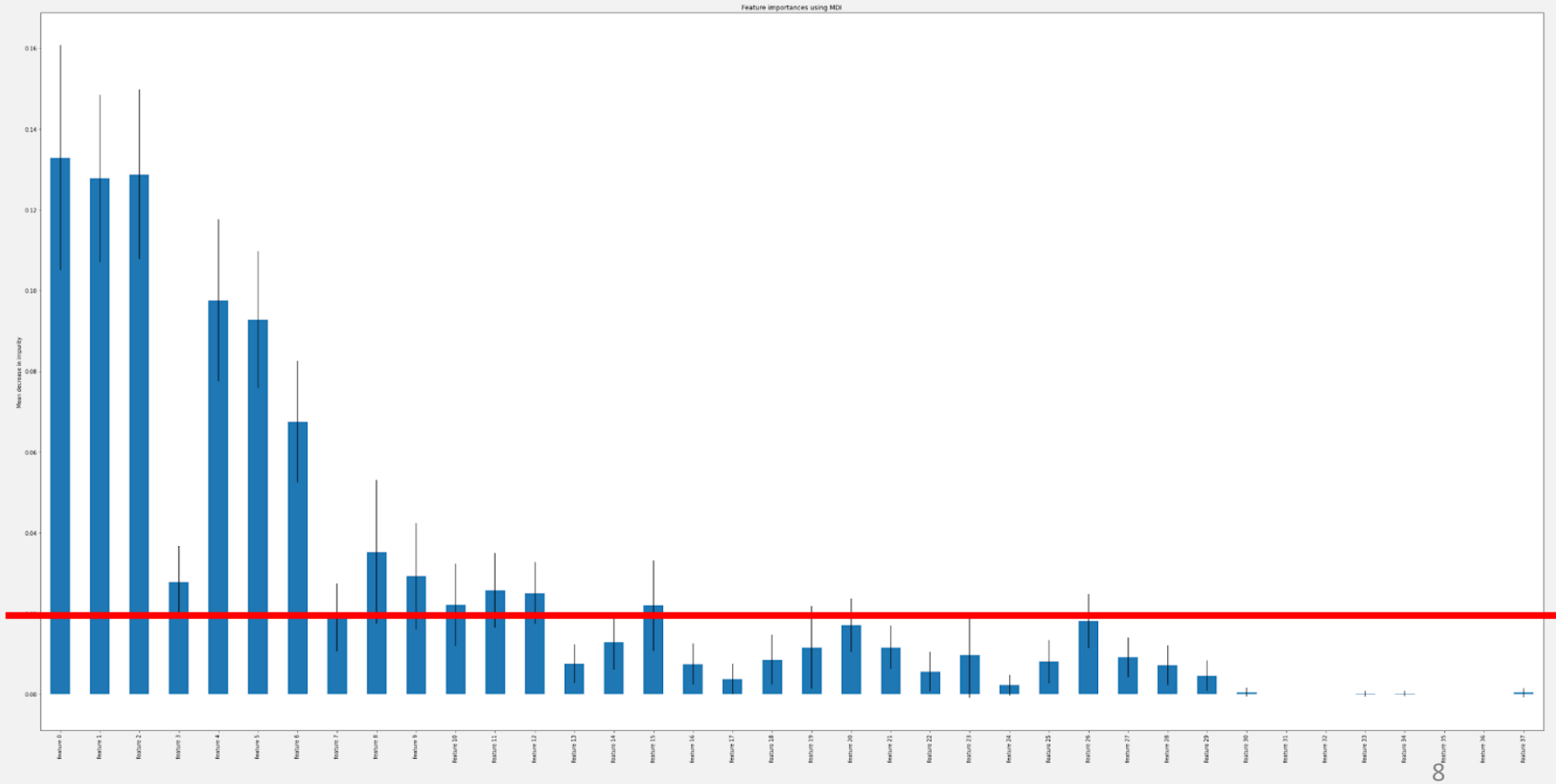
Feature Selection

Numerical

- geo_level_1_id
- geo_level_2_id
- geo_level_3_id
- age
- area_percentage
- height_percentage

Categorical

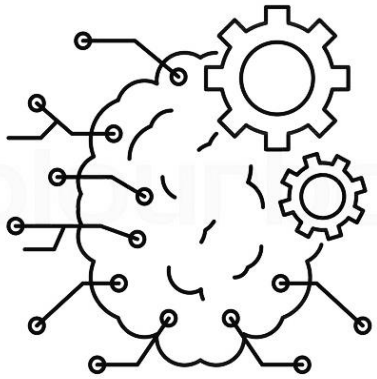
- foundation_type
- groud_roof_type
- other_floor_type
- _floor_type
- position



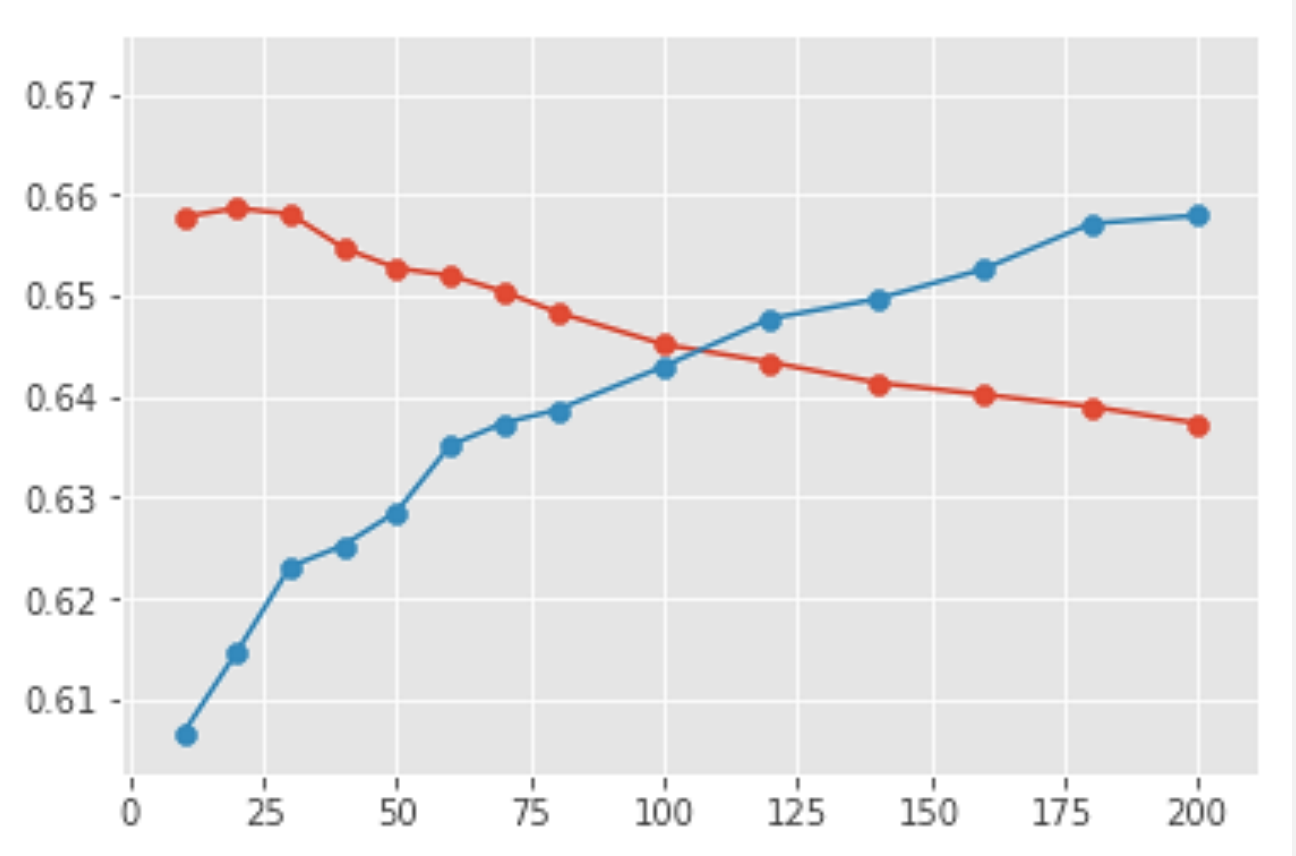
Results – Feature Selection

		Neural Network			
		Before Feature Selection		After Feature Selection	
		Test		Test	
		Test Accuracy	TP3	Test Accuracy	TP3
Case 1	25%	63%	67%	60%	66%
	50%	63%	69%	62%	70%
	75%	64%	75%	65%	71%
Case 2	25%	63%	64%	64%	68%
	50%	63%	66%	65%	70%
	75%	67%	69%	66%	71%

Thank You For Your Attention!



Results



TEAM 11

10:30-10:42

What should you read next?

Building a book recommender

Monday December 6, 2021
Machine Learning 1 (Prof. Golnoosh Farnadi)

Olivier Simard-Hanley, Yifan Yin, Jingyi Zou.

Background

Context:

- Popularity of ebooks and ereaders
- Pandemic : renewed interest in books and sharing reviews online
- Recommender System are crucial to user experience and may increase revenue

Experience:

- Goodbooks10k dataset (2017)
- 6M reviews, 53k users
- 3 files : Books, Ratings, Wishlist

Task :

- Build a recommender system for users with satisfactory performance

Performance

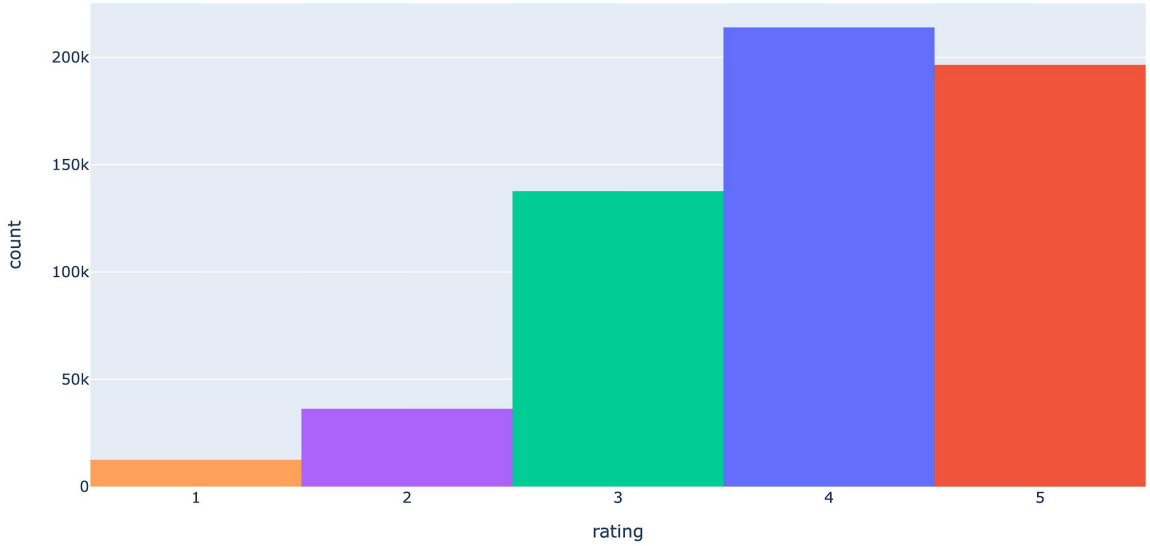
- **How will we evaluate** our models ?
  - **Ground truth = wishlist**
  - We will ask each model to predict 10 books

Performance

1. **Precision**
   - # of wishlist books in the recommendations
2. **Recall**
   - # of recommendations in the wishlist
3. **Binary True positives**
   - If there is at least 1 accurate recommendation
4. **Coverage**
   - What % of the 9958 books are we recommending ?

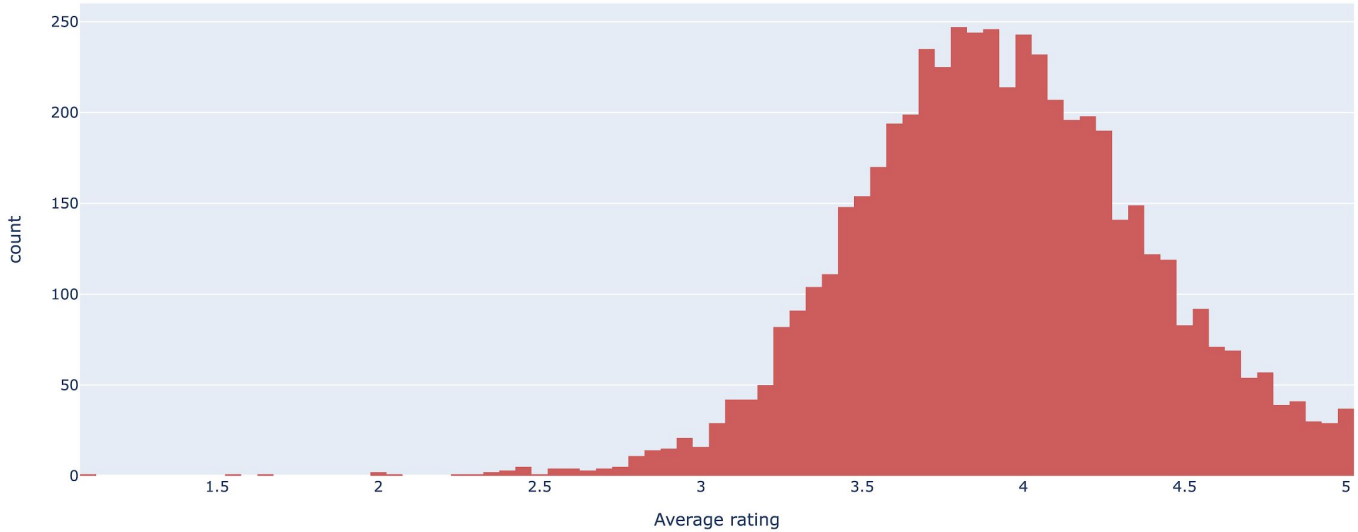
Data Exploration

Star ratings (1 to 5)



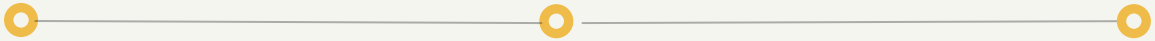
- Most ratings are high
- Could be a problem a recommender system (few examples of unwanted books)

Average rating per user



- Mean rating : 3.92
- Pretty high
- Idiosyncratic
  - we will adjust predictions according to individual rating habits

Process

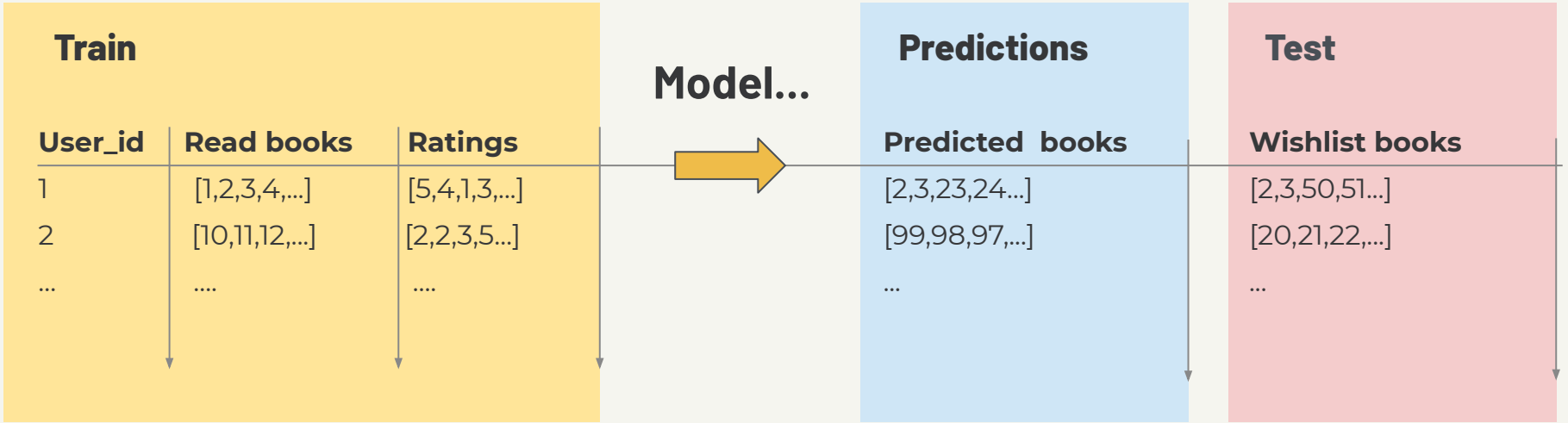


Sampling:
10% of users (5300)
Then selected users with
130+ reviews
Total 1025 users

2 Baselines

3 models

Data structure



Baseline 1 : Linear regression

- **Goal** : predict rating on 5
- **Feature engineering, Scaling, Train-test split**
- **Pros :**
  - Regression is explainable
- **Problems :**
  - 10k+ covariates
  - Speed (sparse matrix > Pandas)
  - **Everyone gets the same recommendation**
  - **Popular books have too much weight**
  - **Interactions have little impact**

Baseline 2 : Popular Recommender

- Recommend 10 most popular books (highest average rating)
- **Pros:**
  - Very simple to implement
  - Could deal with the cold start problem of new users
- **Cons :**
  - **Same recommendations for every user**
  - New books would never be recommended

Better strategies

Tweaked linear Regression

- Adjust ratings according to genre preferences
- Build a **genre preference ratio**
- How likely is it that user **i** would read book **x** ?
- **New rating = predicted rating * ratio**
- For example :
    - Dune (Frank Herbert)
    - Predicted rating of **4.5** with linear regression
    - Ratio is 0.17
    - New predicted rating is **0.77** (4.5 * 0.17)

Tweaked linear Regression

- **Problems :**
  - Long to run
  - Need to build an input matrix for every user
  - Preprocessing is slow (interactions...)

Content-based Filtering

- Compare a book to all other books
- **Text-based features:**
  - “author”, “book_genre” and “book_description”
- **Encode** content with TF-IDF vectorizer
- Build a **similarity matrix** for all pairs of books with cosine similarity

Prediction steps

1. Find a user's **highest rated book**
2. Pick **top 10 similar books**
3. Example: *The Great Dune Trilogy* :
    a. ['Hunters of Dune (Dune Chronicles #7)',
    b. 'Chapterhouse: Dune (Dune Chronicles #6)',
    c. 'The Butlerian Jihad (Legends of Dune, #1)',
    d. 'Heretics of Dune (Dune Chronicles #5)',
    e. 'House Atreides (Prelude to Dune #1)',
    f. ...]
4. Highly explainable

User-based Filtering

- Compare a user to other similar users
- Find vectors of common rated books
- Build a **similarity matrix** for all user pairs with cosine similarity
- Prediction:
    - For a user i, find k-most similar users
    - Pick top rated book of every similar user
- Pro: easy to explain
- Con: cold start problem on new user and new book

Results & Discussion

Final results

Models ⬇️ Averages ➡	Precision	Recall	Binary true positives	Coverage
1. Baseline- LinReg	0.01%	0.0001%	0.1%	7.1%
2. Baseline- Popular	0.05%	0.00048%	0.49%	0.14%
3. Tweaked LinReg	0.39%	0.00381%	3.71%	2.55%
4. Content-based	0.66%	0.00652%	5.72%	37.37%
5. User-based	0.75%	0.00733%	7.12%	2.82%

Potential improvements

- Try hybrid model for solving cold start problem:
  - Popular recommending to new users
  - Content-based model to recommend new books
- Add novelty, serendipity and diversity
  - Recommend a random book in one of the recommendations
  - Might impair the evaluation scores.
- Other evaluation strategies:
  - We did not have time of rating
  - Would have been logical evaluation method
- Try complex models like neural networks

Thanks for listening

Building a book recommender

Monday December 6, 2021
Machine Learning 1 (Prof. Golnoosh Farnadi)

Olivier Simard-Hanley, Yifan Yin, Jingyi Zou.

TEAM 12

10:42-10:54

Customer churn prediction with the hybrid model in the financial institution

Behdad Ehsani

Negar Aminpour

Task and Dataset

What is the task?

Hybrid model

combination of supervised learning and unsupervised learning
(clustering and classification)

1. Increasing the accuracy of the classification models

2. Detecting more probable churners by grouping them

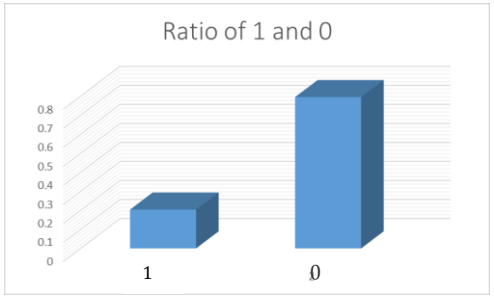
Dataset structure

Socio-demographic data (Gender – Nationality – Age - Tenure – Salary)

Account data (Account balance – Credit score)

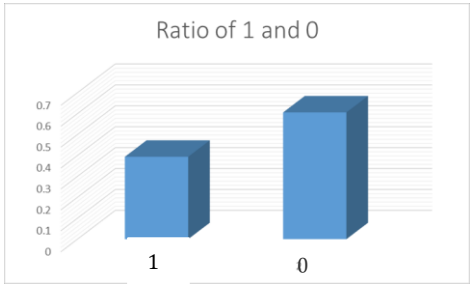
Customer data(Active member? – Number of used bank services? - Has credit card?)

Label? → Churner or non-churner



Dataset is unbalanced

→



Dataset is balanced

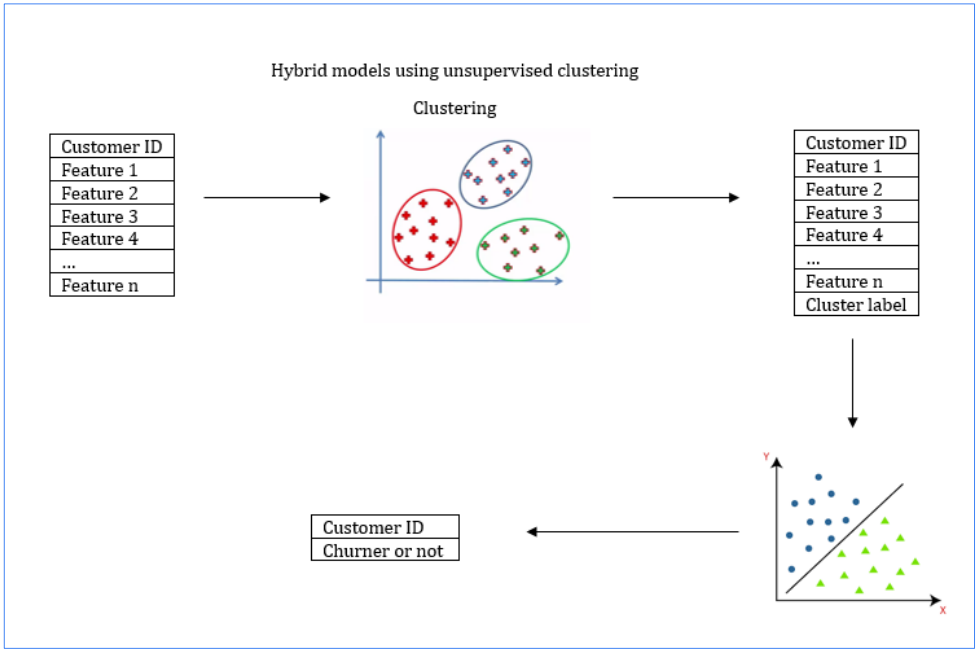
Over-sampling technique

SMOTE MODEL

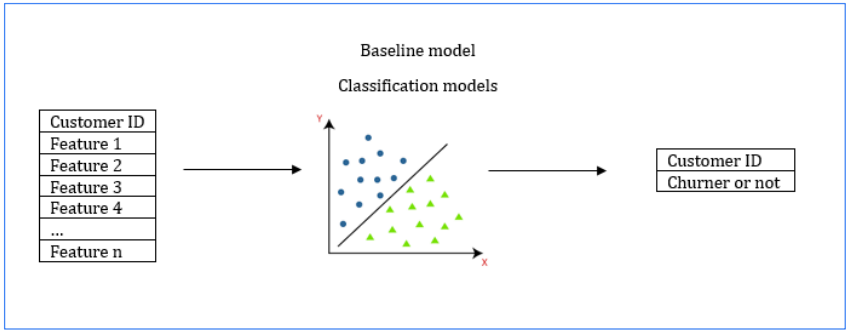
On TRAIN SET

Approach

Hybrid model



Baseline model



Clustering: K-means
Classification: RF-XGB-LR-SVM

Why??
Is Categorizing customers impactful on churn prediction?

Evaluation Setup

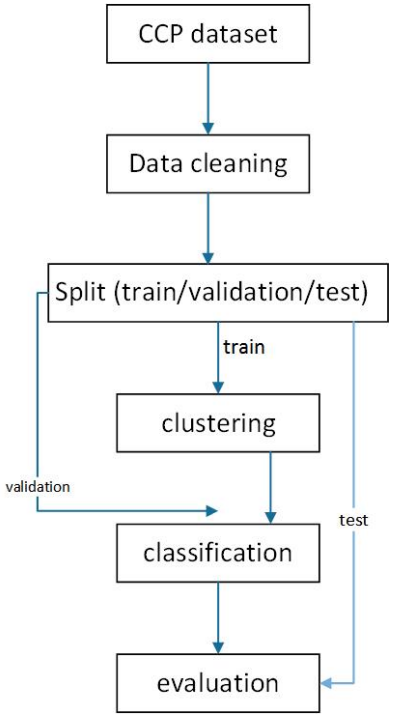
Hyper-parameter?

1. Supervised learning hyper-parameter
2. K (number of clusters) in K-means model

Which supervised learning model?

All of them should be tuned

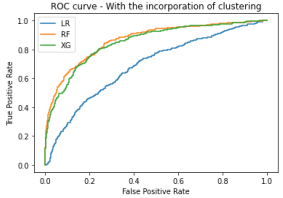
Conducted by Holdout cross-validation



Results - validation

With Clustering - hybrid				
For k = 2				
models	LR	RF	XGB	SVM
accuracy	0.7497	0.8493	0.8343	0.8408
precision	0.3857	0.6289	0.5956	0.5928
recall	0.4165	0.6085	0.5436	0.6608
AUC	0.6868	0.8629	0.8442	-
For K = 3				
models	LR	RF	XGB	SVM
accuracy	0.7513	0.8529	0.8293	0.8383
precision	0.3894	0.6354	0.5789	0.5874
recall	0.4214	0.6259	0.5486	0.6534
AUC	0.6868	0.8673	0.8471	-
For K = 4				
models	LR	RF	XGB	SVM
accuracy	0.7492	0.8493	0.8328	0.8378
precision	0.3858	0.6289	0.5884	0.5873
recall	0.4214	0.6085	0.5561	0.6459
AUC	0.6868	0.8629	0.8432	-
For K = 5				
models	LR	RF	XGB	SVM
accuracy	0.7508	0.8498	0.8388	0.8328
precision	0.3875	0.6272	0.6076	0.5724
recall	0.4165	0.6209	0.5561	0.6608
AUC	0.6863	0.8624	0.8386	-

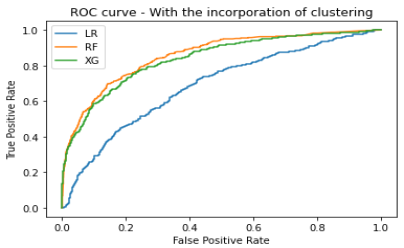
Without Clustering - Baseline				
models	LR	RF	XGB	SVM
accuracy	0.7513	0.8524	0.8303	0.8383
precision	0.3894	0.6366	0.5783	0.5867
recall	0.4214	0.616	0.5711	0.6584
AUC	0.6868	0.8653	0.8503	-



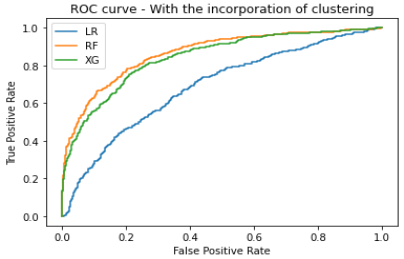
Improvement (in validation set)

But, Result in unseen data (test data) is important

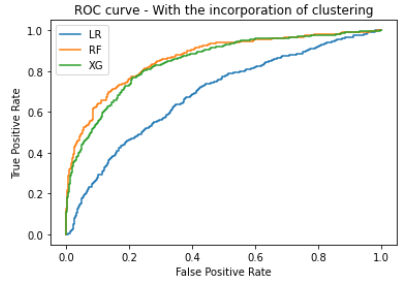
K=5



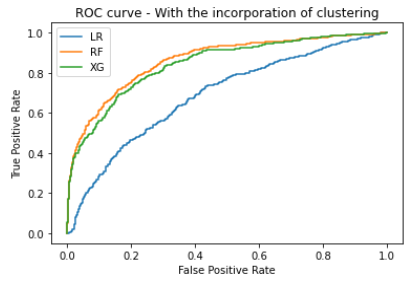
K=4



K=3



K=2



Results - test

Selected hybrid model:

- k=3
- RF
- Max_depth=10
- n_estimators=30

models	RF
accuracy	0.8300
precision	0.5738
recall	0.6796

baseline model:

- RF
- Max_depth=10
- n_estimators=80

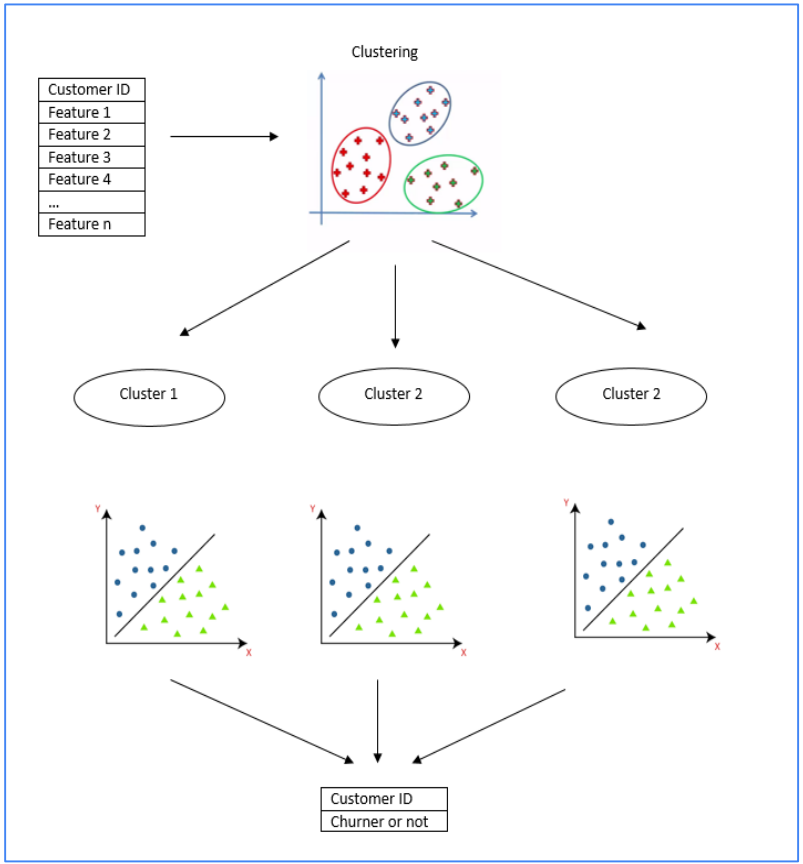
models	RF
accuracy	0.8350
precision	0.5880
recall	0.6650

In test set, the hybrid model does not outperform the baseline model

New approach

Hybrid model - Second version

Classification on each cluster



Results - validation

With Clustering - hybrid				
For k = 2				
models	LR	RF	XGB	SVM
accuracy	0.7528	0.8413	0.8348	0.8303
precision	0.4264	0.5611	0.5586	0.6384
recall	0.3931	0.6148	0.5942	0.5689
For K = 3				
models	LR	RF	XGB	SVM
accuracy	0.7497	0.8378	0.8368	0.8328
precision	0.4364	0.5885	0.5835	0.6409
recall	0.3898	0.5975	0.5954	0.5749
For K = 4				
models	LR	RF	XGB	SVM
accuracy	0.7623	0.8408	0.8268	0.8308
precision	0.4389	0.5711	0.5511	0.6409
recall	0.4131	0.6107	0.5711	0.5698
For K = 5				
models	LR	RF	XGB	SVM
accuracy	0.7618	0.8418	0.8278	0.8303
precision	0.2968	0.5860	0.5586	0.611
recall	0.3802	0.6104	0.5729	0.5724

Without Clustering - Baseline				
models	LR	RF	XGB	SVM
accuracy	0.7513	0.8524	0.8303	0.8383
precision	0.3894	0.6366	0.5783	0.5867
recall	0.4214	0.616	0.5711	0.6584
AUC	0.6868	0.8653	0.8503	-

There is no improvement in validation set

But, Results in test set is important

Results - test

Selected hybrid model:

- k=5
- RF
- Max_depth = 10,15,15,15,15 (for each cluster)
- n_estimators = 60,60,70,20,20 (for each cluster)

models	RF
accuracy	0.8440
precision	0.6699
recall	0.6106

baseline model:

- RF
- Max_depth=10
- n_estimators=80

models	RF
accuracy	0.8350
precision	0.5880
recall	0.6650

In test set, the hybrid model outperform the baseline model

TEAM 13

10:54-11:06

HEC MONTRÉAL

Goodreads Book Recommedation System

Lin Lin, Gurpreet Singh, Lihua Pan

Task and Dataset

What is the task?

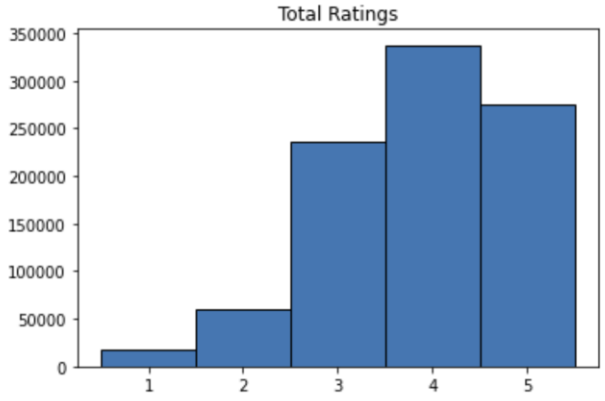
The task is to build a book recommender system based on techniques like cosine similarity and matrix factorization.

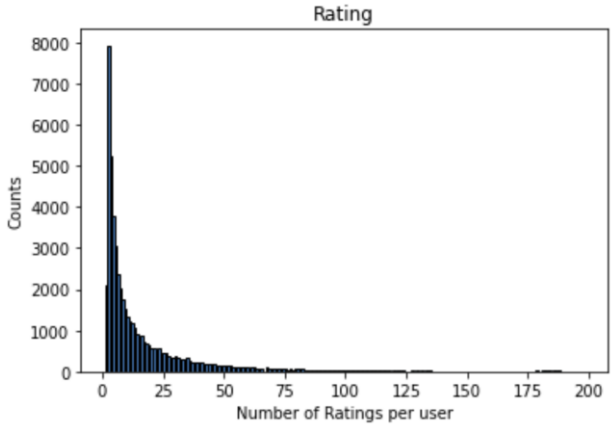
Statistics related to dataset?

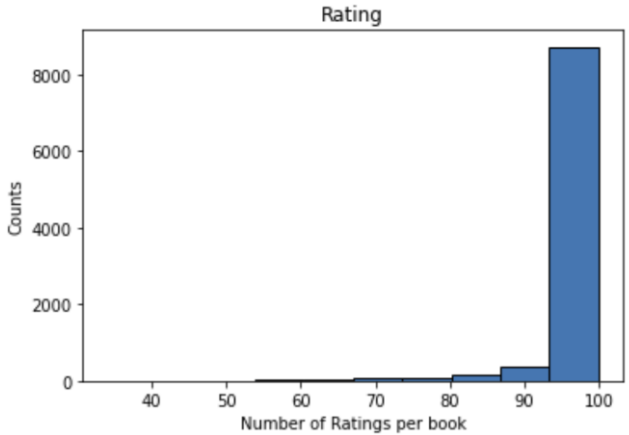
Dataset size: (847700, 3)

Mean	Median	Mode	Variance	Standard Deviation
3.850897	4	4	0.964129	0.981901

Dataset

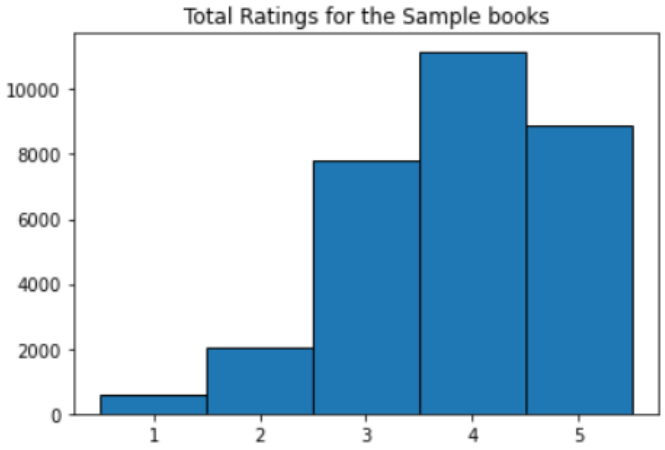

Total Ratings
350000
300000
250000
200000
150000
100000
50000
0
1
2
3
4
5

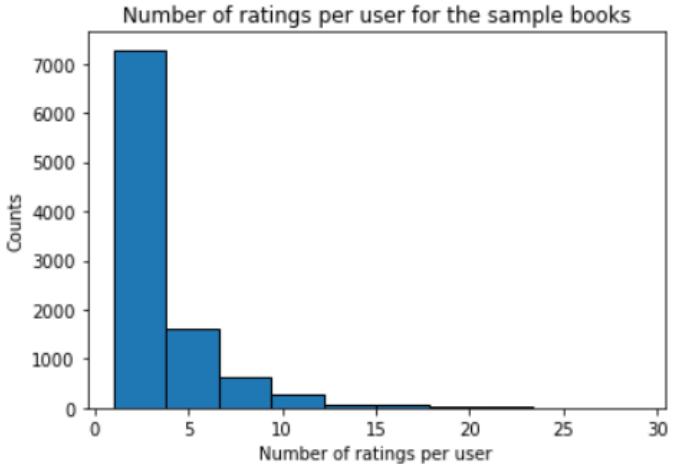

Rating
Counts
8000
7000
6000
5000
4000
3000
2000
1000
0
0
25
50
75
100
125
150
175
200
Number of Ratings per user


Rating
Counts
8000
6000
4000
2000
0
40
50
60
70
80
90
100
Number of Ratings per book

Pre-processing

- Books with no titles are excluded
- Sample 1,000 books and 10,000 users





Collaborative Filtering

Memory Based

- Item-based : Cosine Similarity
- User-based : Cosine Similarity

Algorithm	RMSE
SVD++	0.909
SVD	0.916
NMF	1.066
Random	1.318

Model Based

Matrix Factorization based models using Surprise library.
Attempted algorithm:

- SVD
- SVD++
- NMF
- Random Predictor (baseline)

Cross-validation results (3-fold)

Before hyper-parameter tuning

HEC MONTRÉAL

Collaborative Filtering – Model Based

Hyper-parameter tuning over cross-validation (Grid Search)

Algorithm	Parameter		Best	RMSE
SVD++	No. of factors	50, 75, 100	50	0.898
	No. of iterations	50, 75, 100	75	
	Learning rate	0.005, 0.01	0.005	
	Regularization	0.1, 0.5	0.5	
SVD	No. of factors	25, 50, 100	25	0.899
	No. of iterations	50, 75	75	
	Learning rate	0.005, 0.01, 0.05	0.005	
	Regularization	0.1, 0.5	0.5	
NMP	No. of factors	25, 50, 100	50	1.015
	No. of iterations	25, 50, 75	50	

HEC MONTRÉAL

Evaluation

Training and test data split ratio – 75:25

Algorithm	RMSE
SVD++	0.89
SVD	0.89
NMF	1.01
User-Based (Cosine)	1.89
Item-Based (Cosine)	3.72
Random (Baseline)	1.32

HEC MONTRÉAL

Results

Top 3 books recommended for a user using different algorithms

Algorithm	Model-Based (SVD)		User-Based (Cosine)	
	Book Names	Tags	Book Names	Tags
Top 3 books recommended	Heartburn	fiction; humour; audiobooks	The Art of Travel	paranormal; paranormal romance
	Bad Feminist	Feminism; non-fiction; essay	Desire Unchained (Shadow Lover)	paranormal romance; paranormal
	Essays in Love	philosophy; romance, love, psychology	Ecstasy Unveiled	paranormal romance; paranormal

TEAM 14

11:06-11:18

Predicting the subject of academic papers using ML

Machine Learning for Large-Scale Data Analysis and Decision Making (MATH80629A) – Fall 2021

Timothée Moreau – 11222720

Vladimir Sonin – 11303253

Introduction and Data Investigation

Our goal in this project was to create reliable models that automate the task of multi-label text classification, using Naïve Bayes, K-Nearest Neighbors and Recurrent Neural Networks

Data at a glance

- **arXiv** is a **research-sharing platform open to the public,** run by Cornell University since 1991
- It now hosts nearly **2 million scholarly articles in 8 STEM subject areas**
- For the entry of each paper in the dataset we have the **following information**: ID, Submitter, Authors, Title, Comments, Journal-ref, DOI (Digital Object Identifier), Abstract, Categories and Versions

Focusing on the Abstract of the papers

- The ***abstract*** is a short text that summarises the subject and findings of each paper. We will use this text to **train our models**
- ***Categories*** **will be used as targets** for our predictions. To alleviate running time, we used only 4 out of the 8: **Math, Physics, Comp. Sci and Nuclear Theory**
- There are 1.9 million different articles in our dataset, we have chosen to **work with a subset of 15,733 data points** because of limited resources. The training set contains 70% of the data, the validation set has 20% and the test set 10%

Distribution of Our Data

Among the chosen categories, the sample was taken randomly. Therefore the distribution of our sample should reflect the entire dataset

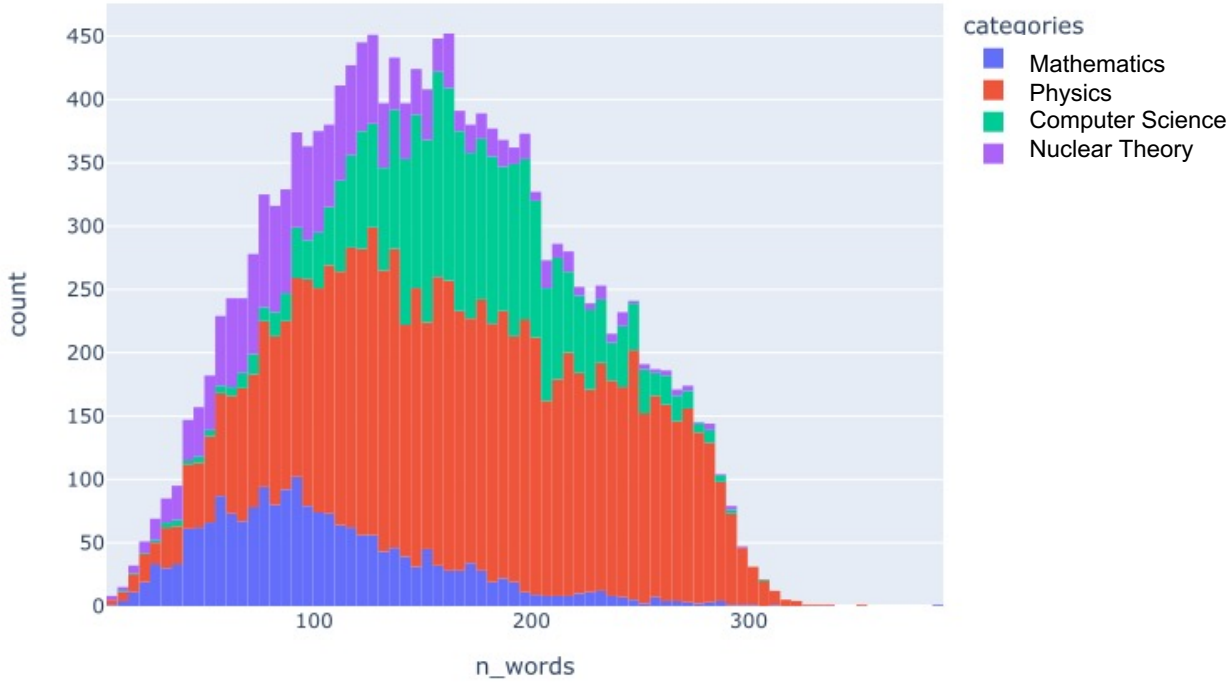
Min: 5 words

Max: 385 words

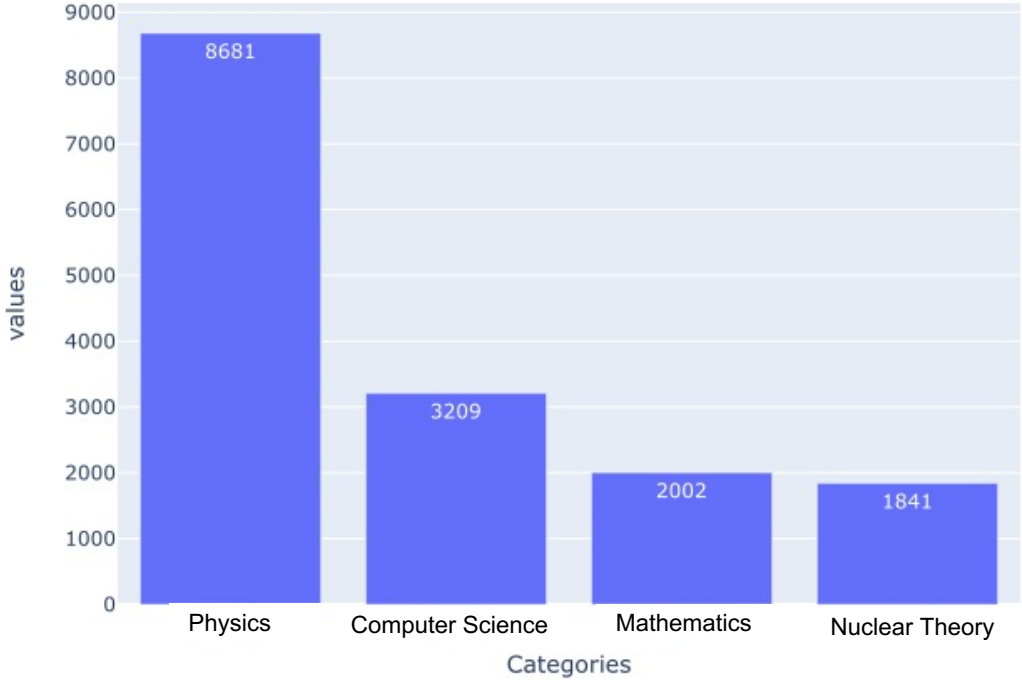
Mean: 154 words

Std. Dev: 66 words

Distribution of Word Count by Abstracts per Category



Number of Data Points per Category



Preprocessing the Data: Keeping What is Relevant for our Model

For our model to take into account only meaningful data, a certain amount of preprocessing was necessary:

1. Remove all special characters and numbers
2. Trim white spaces
3. Lowercase all letters
4. Remove stop words such "a", "the" and "is"
5. Strip accents
6. Pair co-occurring words using n_grams (1, 2)

Before

After

Example

```
In [87]: sample_data['abstract'].iloc[5]
Out[87]: '  We are conducting a transit survey of the open cluster M37 using the Megacam\ninstrument on the 6.5 m Multiple-Mirror Telescope.
We have obtained ~4500\nimages of this cluster over 18.5 nights and have achieved the precision\nnecessary to detect planets smaller than
Saturn. In this presentation we\nprovide an overview of the project, describe the ongoing data\nreduction/analysis and present some of our
preliminary results.\n'

In [88]: sample_data['abstract_tensor'].iloc[5]
Out[88]: 'conducting transit survey open cluster using megacam instrument multiple mirror telescope obtained images cluster nights achieved
precision necessary detect planets smaller saturn presentation provide overview project describe ongoing data reduction analysis present
preliminary results'
```

Model Choice – Training Bag of Words and TF-IDF Using Naive Bayes

Model Choice

> We chose to use the Naive Bayes, KNN and RNN models for this multi-class text classification problem

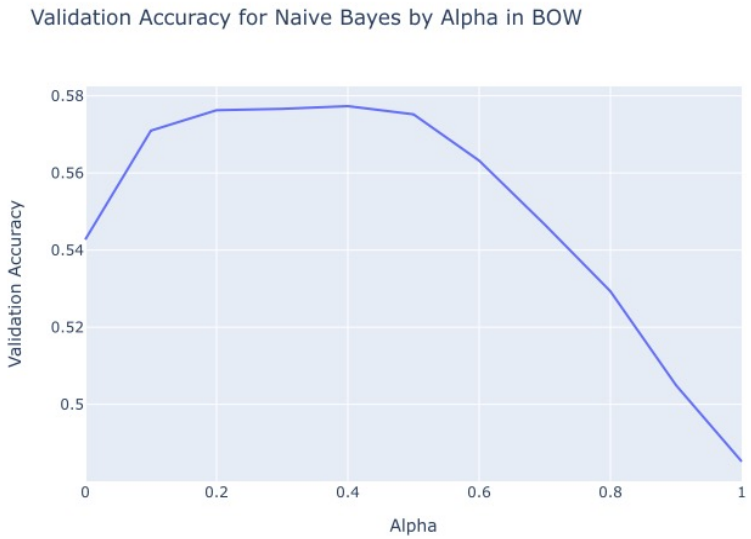
Why Naive Bayes and KNN?

> Suited for multi-class classification
> Fast (NB) and simple to implement
> KNN particularities:
  > No training needed
  > Large real time computation

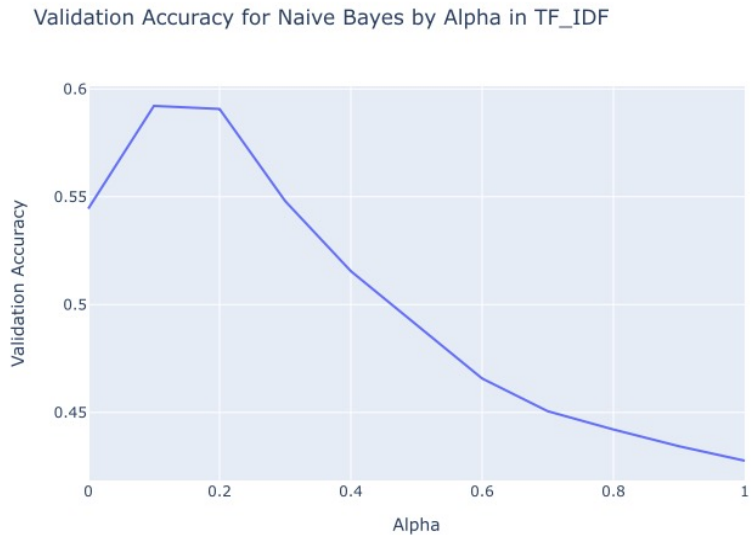
Why RNN?

> More sophisticated and usually has good accuracy
> LSTM allows for a memory of the word sequences, keeping the meaning of sentences
> A lot of options for hyperparameter tuning and architecture (Deep vs Wide)

Bag of Words



TF-IDF



Alpha represents the smoothing parameter

Validation results for BOW:

	precision	recall	f1-score	support
astro-ph	0.611	0.912	0.732	1525
cs.CV	0.733	0.202	0.317	599
math.PR	0.306	0.151	0.202	397
nucl-th	0.326	0.203	0.250	311
accuracy			0.577	2832
macro avg	0.494	0.367	0.375	2832
weighted avg	0.563	0.577	0.517	2832

Validation results for TF_IDF:

	precision	recall	f1-score	support
astro-ph	0.603	0.936	0.734	1525
cs.CV	0.690	0.285	0.404	599
math.PR	0.349	0.096	0.150	397
nucl-th	0.370	0.129	0.191	311
accuracy			0.592	2832
macro avg	0.503	0.362	0.370	2832
weighted avg	0.560	0.592	0.523	2832

KNN and RNN Hyperparameter Tuning

KNN

Hyperparameters:
Training 500 training and 100 validation
Average Experiment time ~4 Hour/K

K-Neighbors	Validation Accuracy
1	0.79
3	0.81
5	0.79
7	0.78

RNN

Hyperparameters:
1000 Abstracts in training set, 200 abstracts in validation set, 10 Epochs, batch size=32
Average Experiment time ~1 Hour
First Activation layer = 'relu'; Last Activation = 'softmax'
*Other hyperparameters have been tested with little variability of accuracy (learning rate, batch size, # nodes etc.)

Wider Model

Layer 1	Layer 2	Layer 3	Train Accuracy	Validation Accuracy
LSTM(256)	Dense(64,4)		0.75	0.46
LSTM(128)	Dense(32,4)		0.72	0.53
LSTM(128)	LSTM(64)	Dense(32, 4)	0.71	0.54
LSTM(128)	Dense(64,4)		0.67	0.57
LSTM(128)	Dense(64,4, Dropout =0.1)		0.65	0.51
LSTM(64)	Dense(32,4)		0.63	0.56
LSTM(128)	Dense(64,4, Dropout =0.5)		0.55	0.45

Overfitting

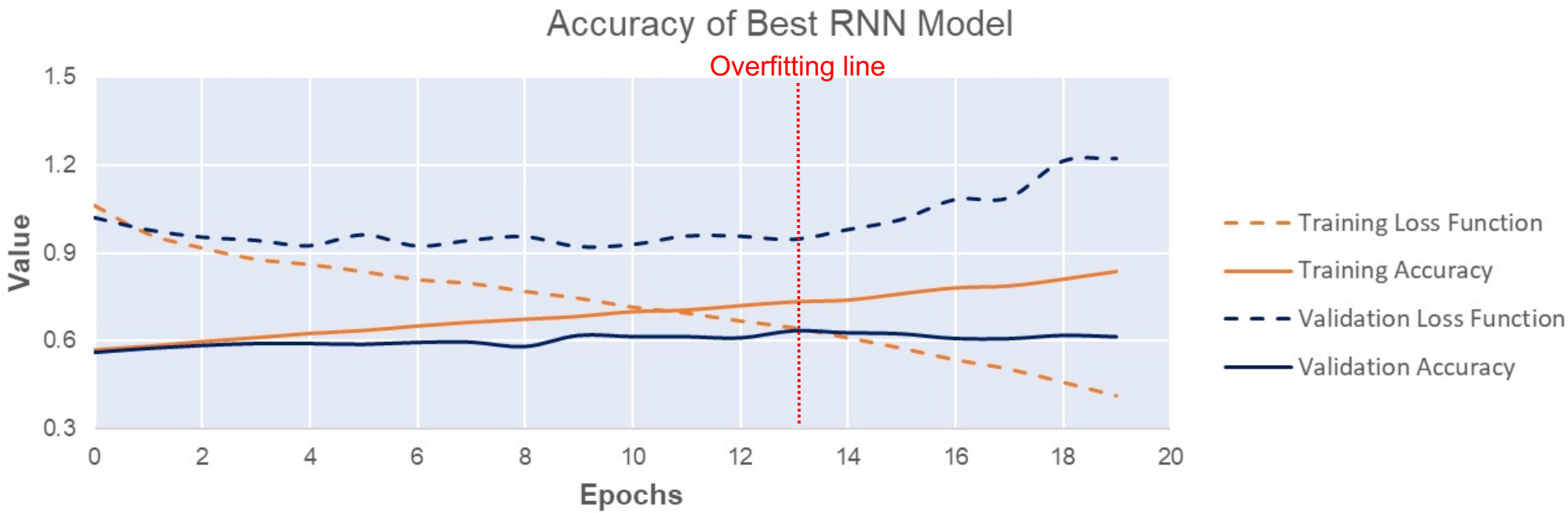
Right fit

Underfitting

Deeper Model

Accuracy of Chosen Model Over 20 Epochs

Architecture: 1st Layer: LSTM(128), 2nd Layer: Dense(64,4)
Number of training abstracts: 11,013
Number of validation abstracts: 3,147
13 Epochs was found to be ideal for our given model



Results – Which Model Gave the Best Accuracy for our Test Set?

Models	Training	Validation	Test
Baseline (Astro-Physics)	55.5%	53.8%	55.3%
BOW Naïve Bayes	98.8%	57.7%	47.2%
TF-IDF Naïve Bayes	99.3%	59.2%	51.1%
KNN	-	81%	88%
RNN	67%	57%	60.4%

- Both Naive Bayes method have very high training accuracy, a somewhat good validation accuracy relative to other tests but a **very poor performance on the test set**. This is a sign of **overfitting** to the training data
- **RNN performs better** than Naive Bayes and Baseline, but its **accuracy remains modest**
- The RNN architecture **shows that deeper and wider models aren't always the best**. We need to strike the right balance
- **KNN performs the best** of all models, both in terms of validation and test accuracy

Difficulties encountered and future improvements

Technical difficulties

- Not properly inspecting the quality of inputs to our model, and getting very bad accuracy initially
- Having abstracts with very similar subjects, making it hard to distinguish between them
- Setting up software for GPU computation to increase speed of computations
- Understanding hardware bottlenecks

Improvements

- Extend classification problem to all 8 categories, possibly more for other similar problems
- Do more hyper-parameter tuning and additional grid searches to find optimal model
- Allocate more time in preprocessing (increase StopWord list)
- Implement parallel computing to reduce running time
- Try additional RNN architectures or BERT

Conclusion

Questions and Answers

TEAM 15

11:18-11:30

Predicting User Unsubscribing Behavior: A Multimodal ML Approach

Team Neural Niche:

Syed-Muhammad-Usman Tayyab, Mitchell Karkheck, Yinte Zhang

Problem Statement:

- Predict whether mobile financial app users will UNSUBSCRIBE
- Through multiple ML predictive models
- Compare the performance

- Why? Assist in building guidelines to minimize the unsubscribing rate and maximize customer retention

Research Questions

- **RQ1:** Which features of user behaviors are most important in predicting the label? (i.e. unsubscribe/stay subscribed)
- **RQ2:** Which Model performs better in predicting the label?

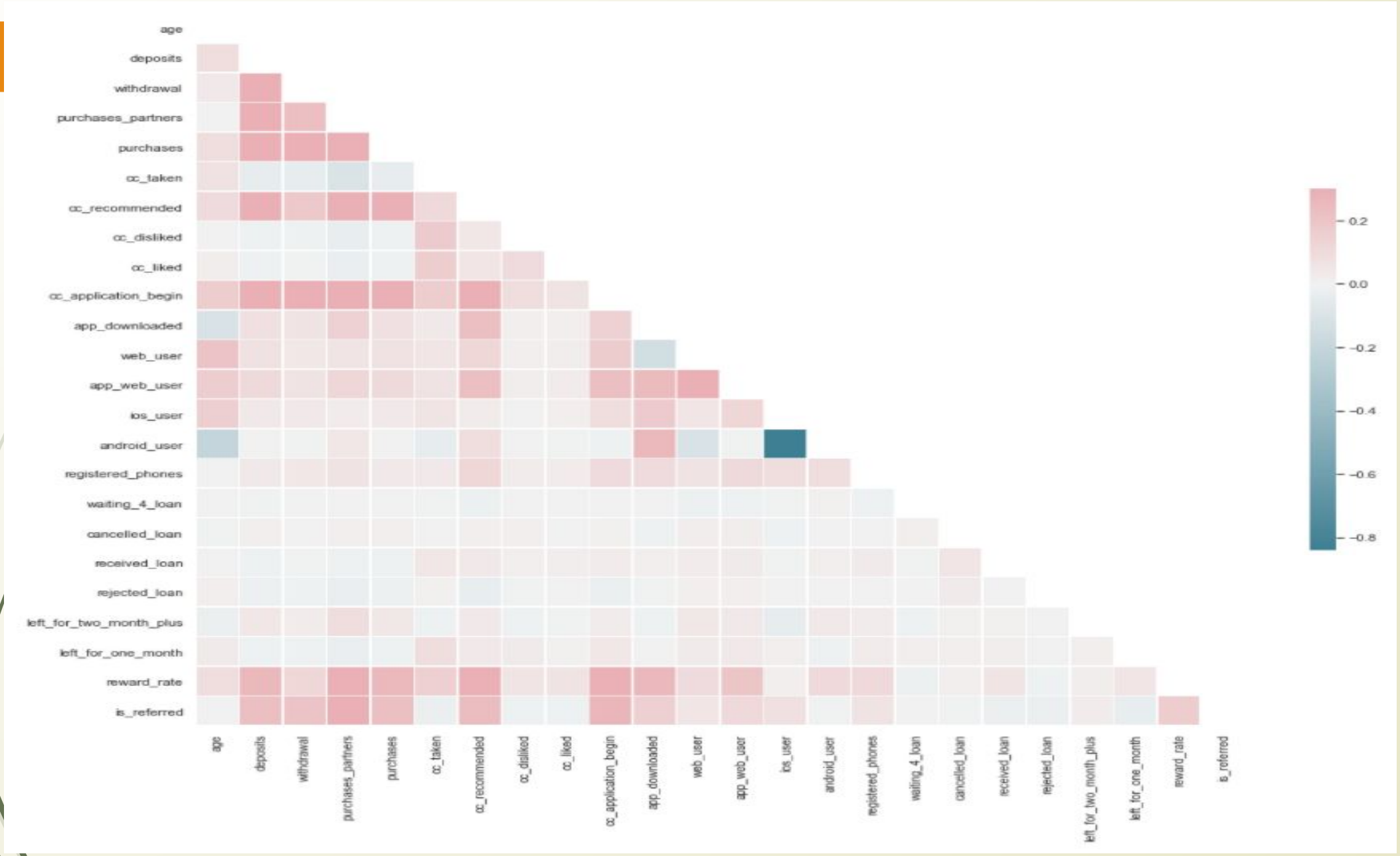
- Task (T)
  - Binary Classification
  - 1 if unsubscribe, 0 otherwise
- Performance measure (P)
  - ROC-AUC
- Experience
  - Supervised Learning
    - use >20 features and 1 label to fit a model
    - use the model to make predictions on labelled test set

Data

- Almost 27000 data points
- 29 features including Age, Credit Score, Housing Type, etc…
  - categorical
  - numerical
  - binary
- Target Variable: Unsubscribing Rate (churn)

Data: Features

- user
- churn - (target)
- age
- housing
- credit_score
- deposits
- withdrawal
- purchase_partners
- purchases
- cc_taken
- cc_disliked
- cc_liked
- cc_application_begin
- app_downloaded
- web_user
- app_web_user
- android_user
- registeded_phones
- payment_types
- waiting_4_loan
- cancelled_loan
- recieved_loan
- rejected_loan
- zodiac_sign
- left_for_two_month_plus
- left_for_one_month
- rewards_earned
- reward_rate
- is_referred


age
deposits
withdrawal
purchases_partners
purchases
cc_taken
cc_recommended
cc_disliked
cc_liked
cc_application_begin
app_downloaded
web_user
app_web_user
ios_user
android_user
registered_phones
waiting_4_loan
cancelled_loan
received_loan
rejected_loan
left_for_two_month_plus
left_for_one_month
reward_rate
is_referred
0.2
0.0
−0.2
−0.4
−0.6
−0.8

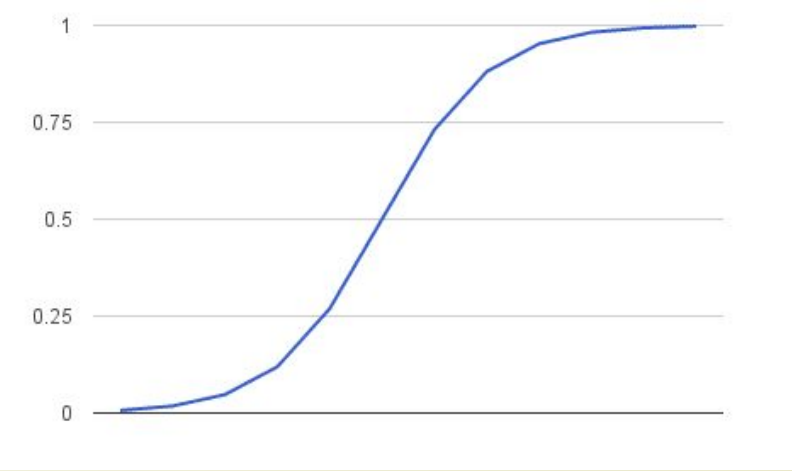
Models

1. Logistic Regression
2. Support Vector Machine
3. AdaBoost Classifier
4. XGBoost Classifier
5. Light GBM Decision Tree

} Deleted two variables b/c too many NAs

} Kept all variables

Logistic Regression Classifier

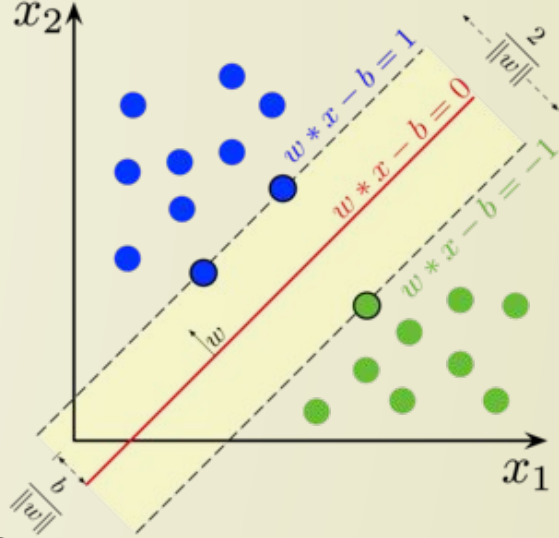


- Most commonly used when the target variable is binary.
  - output: P(y=1 | X)
- Loss Function
  - Logistic Loss

$$Cost(h_\theta(x), y) = -ylog(h_\theta(x)) - (1 - y)log(h_\theta(x))$$

- Regularization
  - L1 and L2 Regularization
- Simple and Traditional classification model

Support Vector Machine

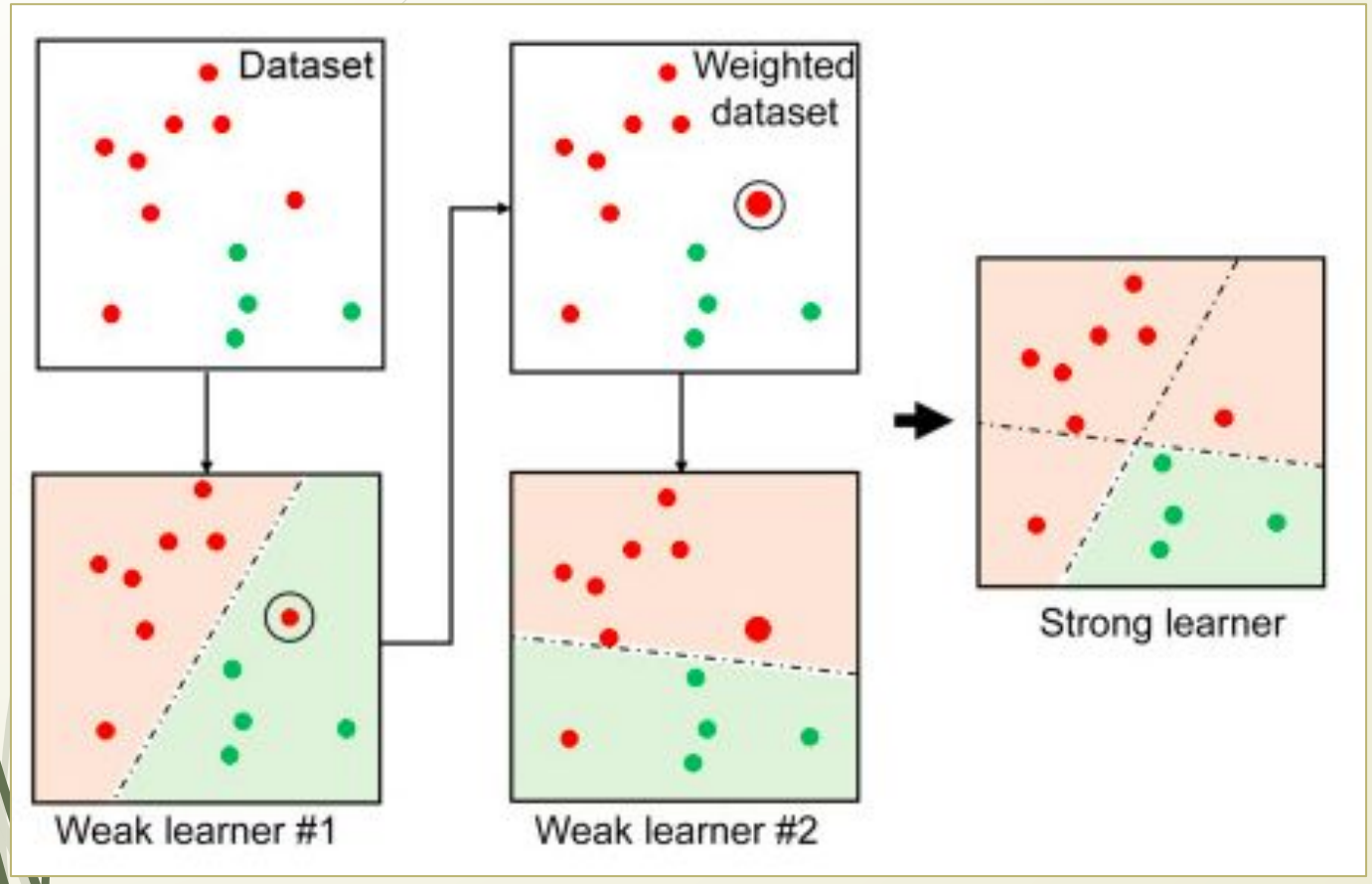


- Linear Classifier
- Trade-off between maximizing the margin and minimizing the training error
- Objective: maximize the margin, while softly penalizing outliers

$$\min_{w} C \sum_{n=1}^{N} \mathcal{E}_n + 0.5||w||^2$$

- The penalty term **C** controls the strength of this penalty, and acts as an inverse regularization parameter
- Can map the input data to a higher dimensional space to assign a decision boundary

Adaboost Classifier



Hyperparameters:

1. Number of weak learners
2. Depth of weak learners
3. Learning rate

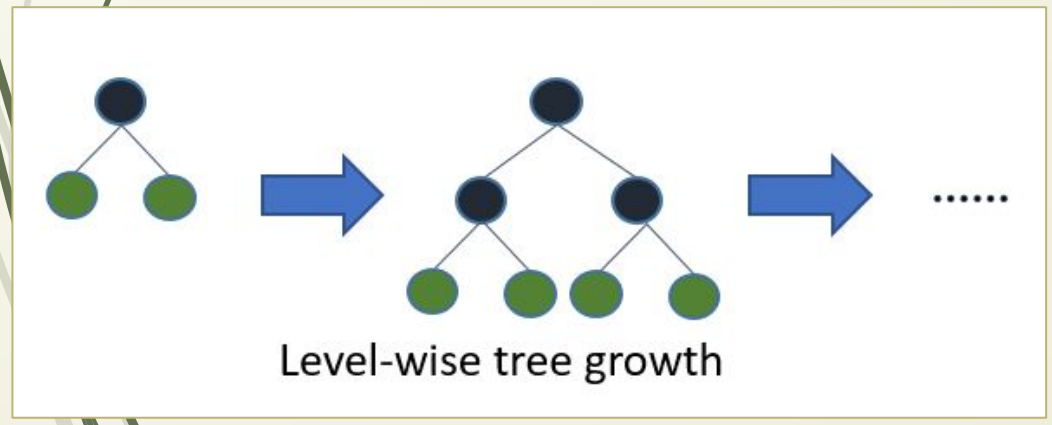
Note: trade-off between 1 and 3

Loss function:

$$L = \frac{1}{N}\Sigma_{i=1}^{N} e^{-y_i f(x_i)}$$

XGBoost: Extreme Gradient Boosting

- Tree based algorithm, boosting weak learners using the gradient descent architecture
- XGBoost improves on the basic GBM framework by optimizing and enhancing the algorithm
- System optimization is achieved through parallelization, tree pruning and hardware optimization
- Algorithmic enhancements include regularization, sparsity awareness, weighted quantile sketches and cross-validation

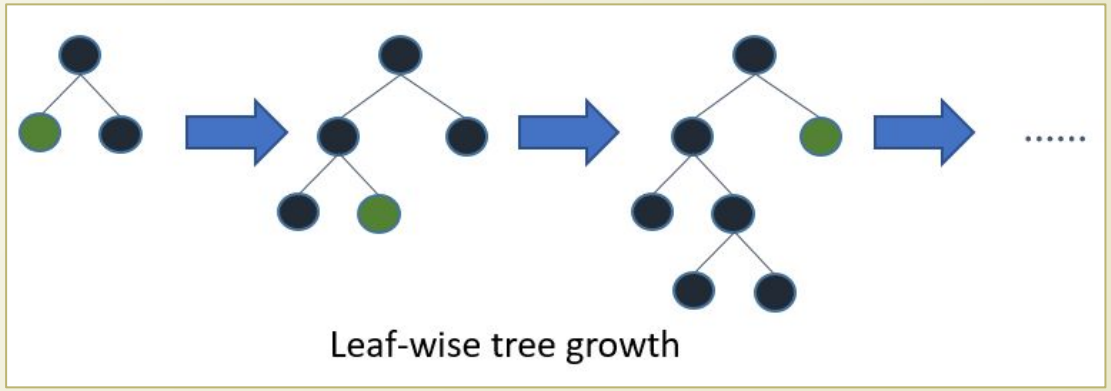


XGBoost: Hyperparameters

- **Max Depth**: Maximum depth of a tree. Increasing makes model more complex and likely to overfit
- **Min child weight**: Minimum sum of instance weights needed in a child. Corresponds to minimum number of instances needed per node.
- **Subsample**: Subsample ratio of the training instances
- **colsample_bytree**: subsample ratio of columns when selecting each tree
- **ETA**: Learning Rate or step size shrinkage used in update to prevent overfitting
- Parameters were tuned to maximize validation AUC

Light GBM

- a fast, distributed, high-performance gradient boosting framework based on decision tree algorithm
- Similar to XGB, but tree growth is leaf-wise - chooses the split point that has the largest gain



Leaf-wise tree growth

- Objectives
  - Binary classification --- Binary log loss function

Light GBM Hyperparameters

- For best fit
  - **Max depth**: It specifies the maximum depth or level up to which tree can grow. It effects model complexity.
  - **Number of leaves**: the number of leaves to be formed in a tree. Number of leaves must be set to a smaller value otherwise it may lead to overfitting.
  - **Min data in leaf** : It is also one of the important parameters in dealing with overfitting. Setting its value smaller may cause overfitting and hence must be set accordingly.

- For faster speed
  - **Bagging fraction:** Is used to perform bagging for faster results
  - **Feature fraction:** Set fraction of the features to be used at each iteration

Results

Logistic Regression Classifier

Training Results

ACCURACY SCORE: = 0.6499

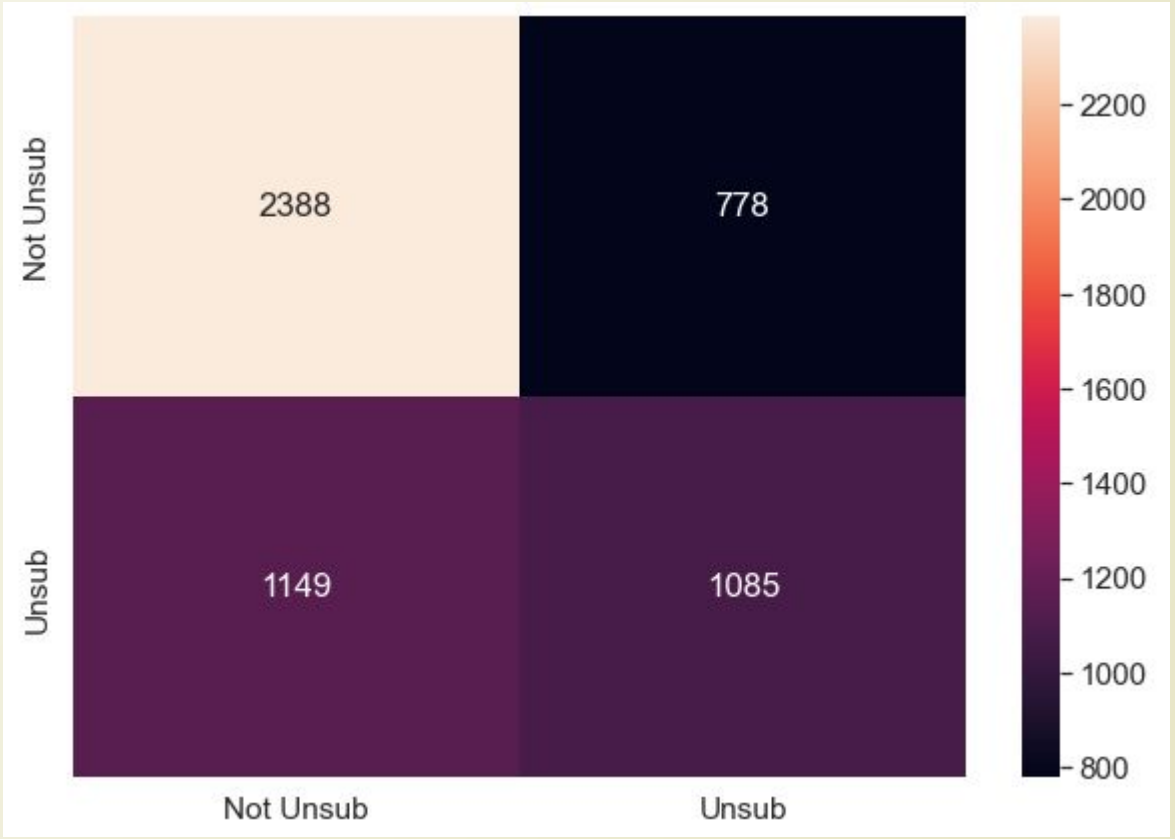
ROC-AUC SCORE: = 0.6268

Test Results

ACCURACY SCORE: = 0.6420

ROC-AUC SCORE: = 0.6186

Confusion Matrix (Test Accuracy results)



Support Vector Machine

Training Results

ACCURACY SCORE: = 0.7202

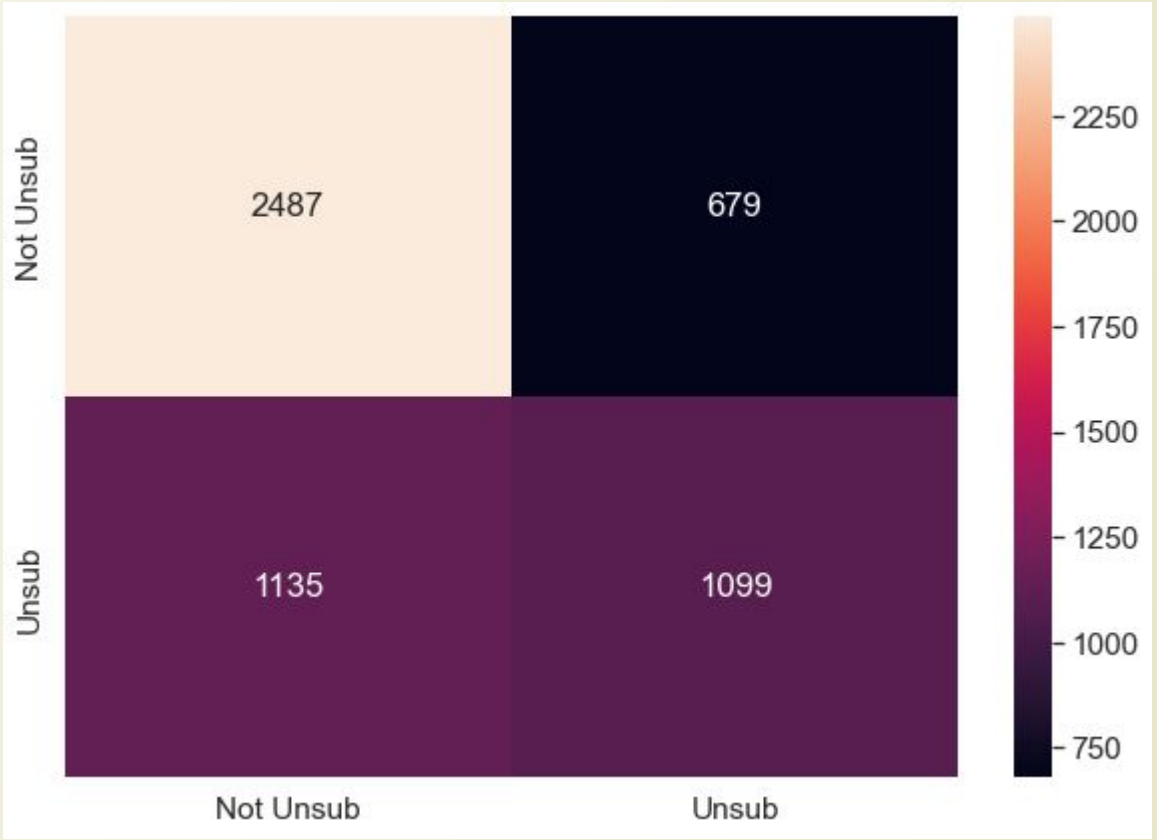
ROC-AUC SCORE: = 0.6966

Test Results

ACCURACY SCORE: = 0.6641

ROC-AUC SCORE: = 0.6387

Confusion Matrix (Test Accuracy results)



AdaBoost Classifier

Training Results

ACCURACY SCORE: = 0.7715

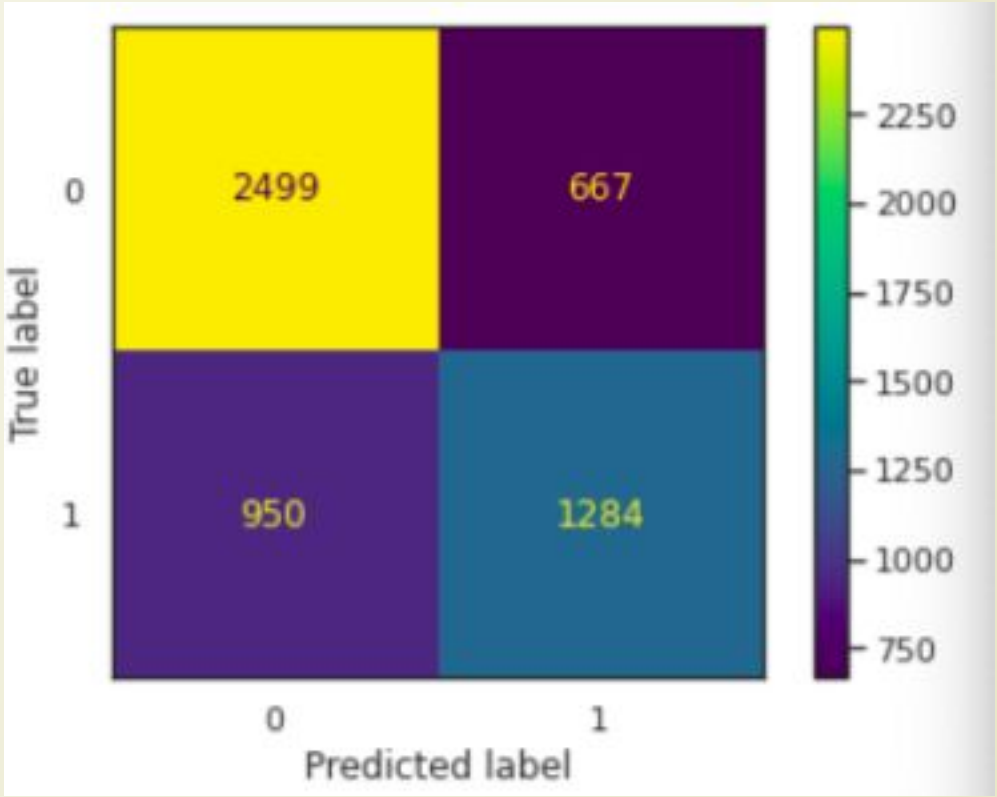
ROC-AUC SCORE: = 0.7569

Test Results

ACCURACY SCORE: = 0.7006

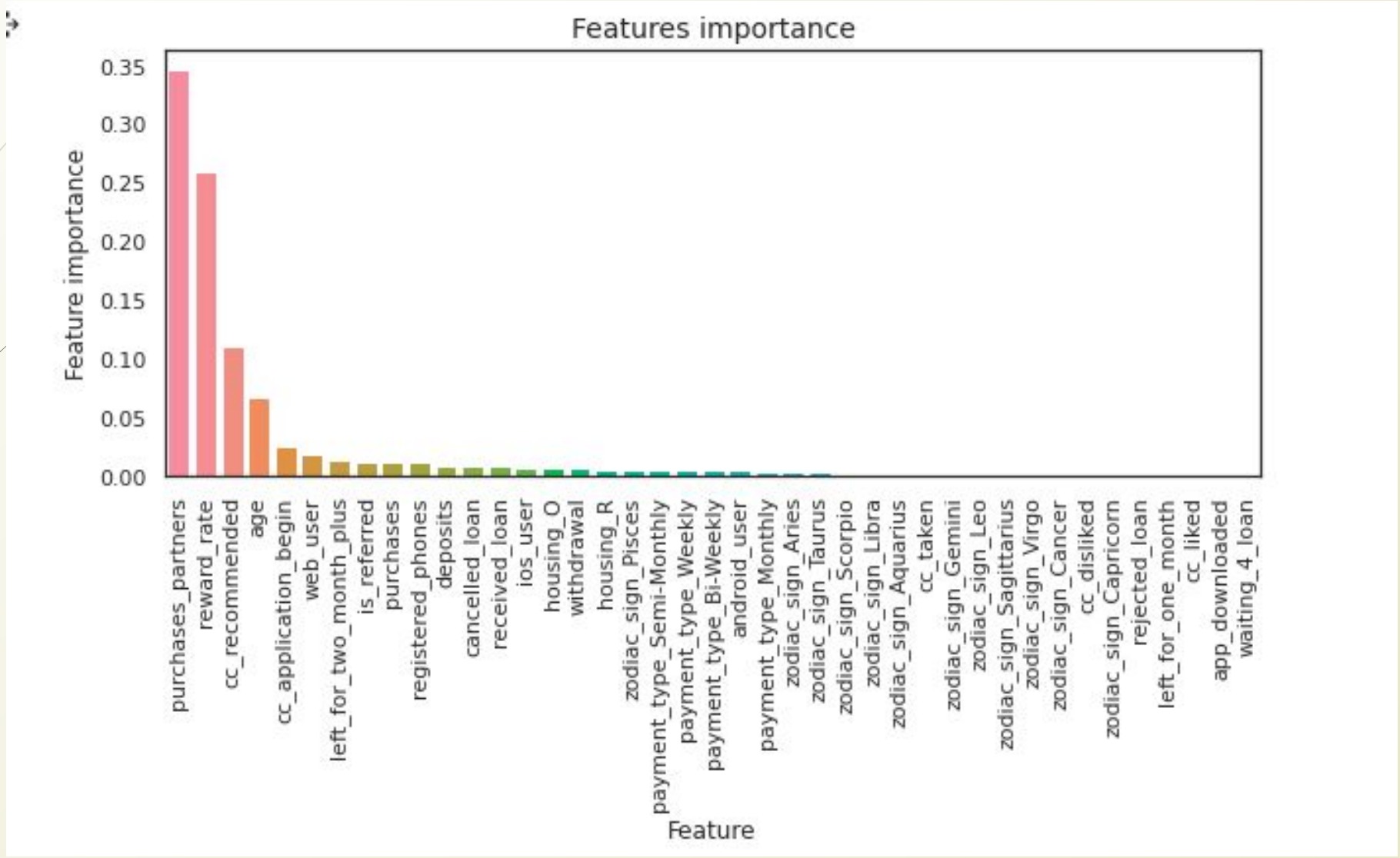
ROC-AUC SCORE: = 0.6820

Confusion Matrix (Test Accuracy results)



Predicted Values

AdaBoost Classifier



XGBoost Classifier

Training Results

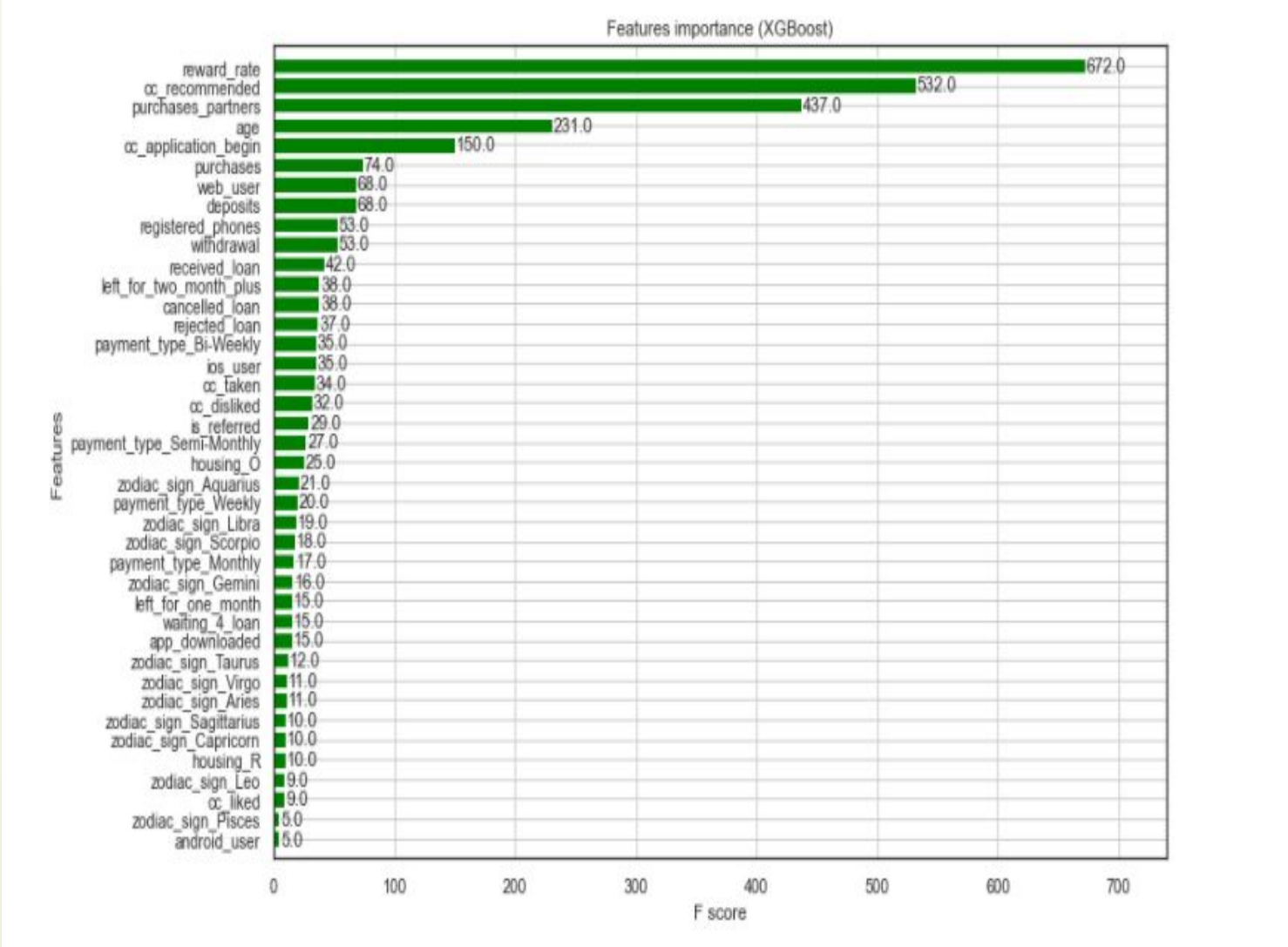
ROC-AUC SCORE: = 0.9382

Validation Results

ROC-AUC SCORE: = 0.77202

Test Results

ROC-AUC SCORE: = 0.7845



Light GBM Decision Tree

Training Results

Avg precision score = 0.8076

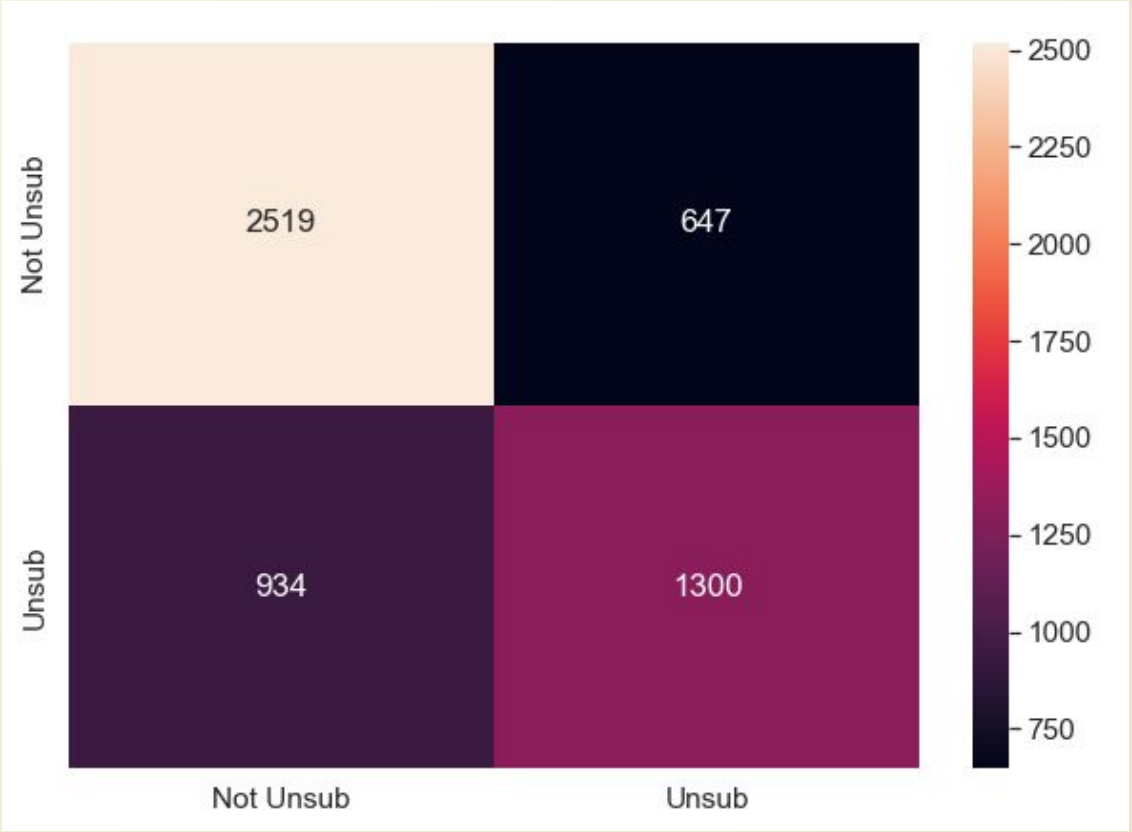
ROC-AUC SCORE = 0.8587

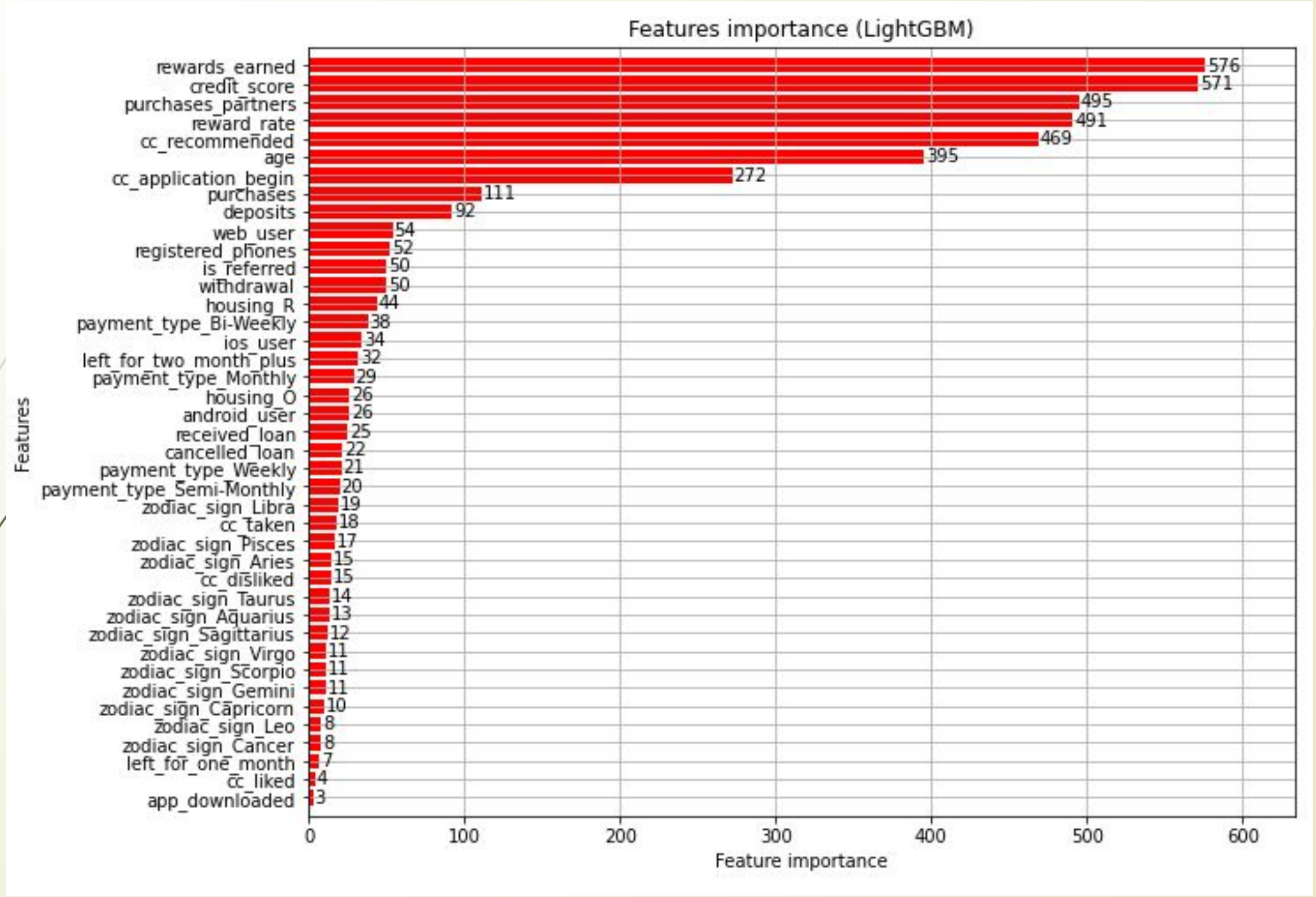
Test Results

Avg precision score = 0.7123

ROC-AUC SCORE = 0.7774

Confusion Matrix (Test Accuracy results)

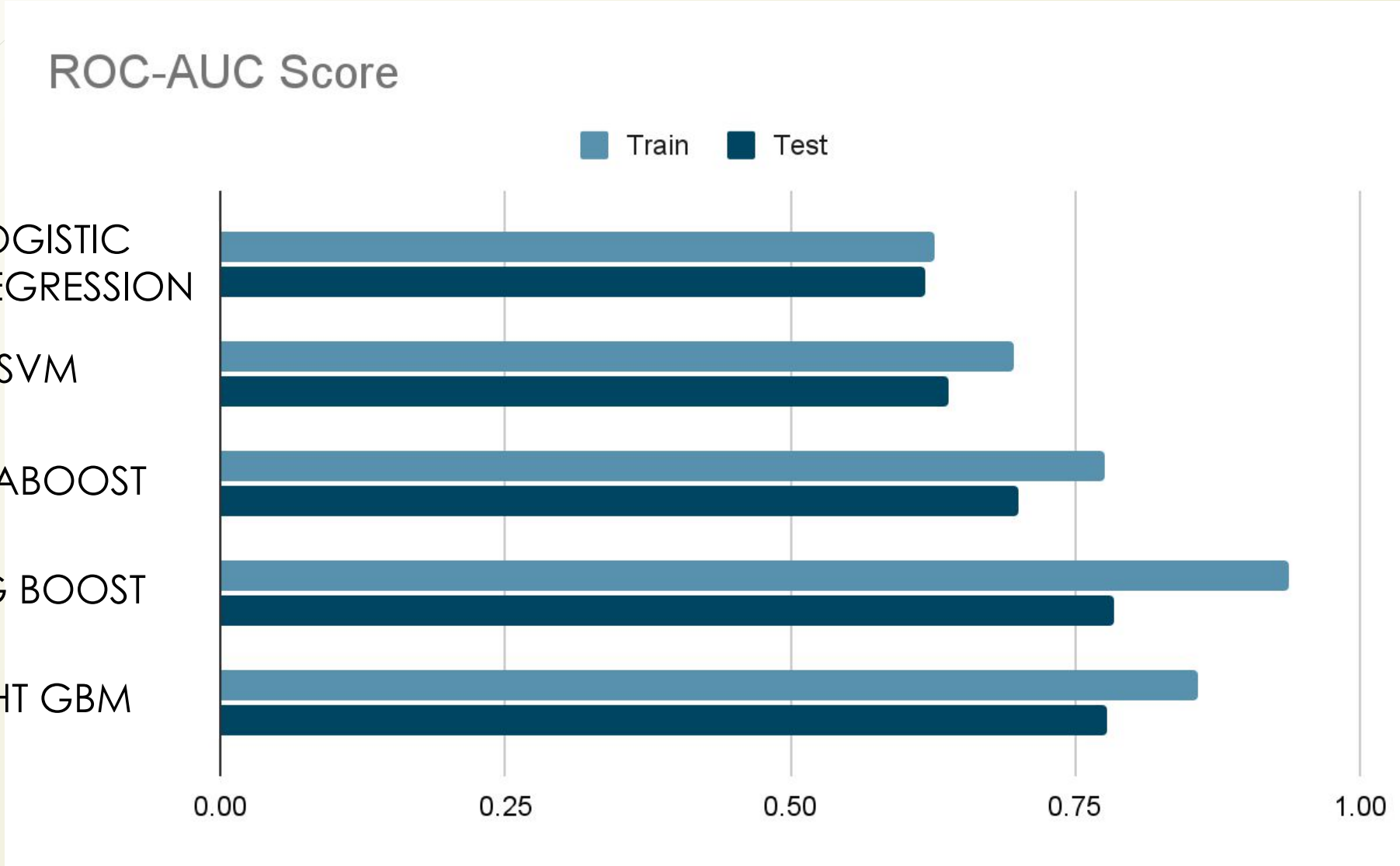



Features importance (LightGBM)
rewards_earned 576
credit_score 571
purchases_partners 495
reward_rate 491
cc_recommended 469
age 395
cc_application_begin 272
purchases 111
deposits 92
web_user 54
registered_phones 52
is_referred 50
withdrawal 50
housing_R 44
payment_type_Bi-Weekly 38
ios_user 34
left_for_two_month_plus 32
payment_type_Monthly 29
housing_O 26
android_user 26
received_loan 25
cancelled_loan 22
payment_type_Weekly 21
payment_type_Semi-Monthly 20
zodiac_sign_Libra 19
cc_taken 18
zodiac_sign_Pisces 17
zodiac_sign_Aries 15
cc_disliked 15
zodiac_sign_Taurus 14
zodiac_sign_Aquarius 13
zodiac_sign_Sagittarius 12
zodiac_sign_Virgo 11
zodiac_sign_Scorpio 11
zodiac_sign_Gemini 11
zodiac_sign_Capricorn 10
zodiac_sign_Leo 8
zodiac_sign_Cancer 8
left_for_one_month 7
cc_liked 4
app_downloaded 3
Features
0
100
200
300
400
500
600
Feature importance

Model Comparison

Model	With Default settings ROC-AUC SCORE		With Hyperparameter tuning/ regularization ROC-AUC SCORE		ROC-AUC SCORE with Additional Features	
	Train	Test	Train	Test	Train	Test
Logistic Regression	0.6274	0.6182	0.6268	0.6186	--	--
Support Vector Machine	0.6966	0.6387	0.6966	0.6387	--	--
AdaBoost Classifier	0.6599	0.6471	0.7569	0.6820	0.7759	0.7004
XGBoost Classifier	0.7773	0.7481	0.90554	0.76355	**0.9382**	0.7845
Light GBM Decision Tree	0.7426	0.6888	0.8290	0.7584	0.8587	0.7774

Model Comparison



Conclusions

- All 5 models worked better than the random baseline of 50%
- Tree-base boosting Models performed considerably better than the conventional classification models
- By tuning the modern models, their performance can be considerably improved, but they have a tendency to overfit the training data.
- All models have similar feature importance
  - credit score
  - rewards earned
  - purchases partners