Machine Learning for Large-Scale Data Analysis and Decision Making (MATH80629A) Winter 2022

Team 1

HEC MONTRÉAL

Predicting Bus Travel Time Delay and Earliness Using Machine Learning Techniques

Melina Asadi, Mohammad Ghavidel

6/April/ 2022

Agenda

1. Introduction
2. Task and Dataset
3. Literature Review
4. Solution Approach
    1. Evaluation Setup
    2. Results
5. Future Direction

Introduction

- ✓Buses travel time delay and earliness arrival are the major reasons for people to use them as a transportation mode.
- ✓Such issues will push passengers to use personal cars rather than buses that would lead to more traffic congestion, more fuel consumption and air pollution.
- ✓We aim to predict such delays and earliness for New York City's bus transit system regarding different attributes such as Distance to the next stop, Weather description, Humidity, Temperature, Wind Speed, Schedule arrival time and Traffic condition.

HEC MONTRÉAL

Task and Dataset

What is the task?

- Classification (Supervised Learning)

The main goals of this study are:

1. What is the most important attribute for bus delays?
    - Finding the most important attributes which have high correlations with buses' travel time delay and earliness arrival.
2. Is the bus reliable (delay/on-time) commutating mode?
    - Predicting the occurrence of buses travel time delays (delayed or not delayed), regarding the scheduled and actual arrival time of buses.

HEC MONTRÉAL

Task and Dataset

- ✓The data obtained from Bus Breakdown and Delays and Historical Hourly Weather Data.
- ✓Containing the Traffic condition, departure time , distance to the next stop from bus stop and delays from Aug 2017.
- ✓Weather Description, Humidity, Temperature and Wind Speed from Historical Hourly Weather Data.
- ✓Each of the mentioned datasets have several attributes and based on the most seen attributes in literature, most frequent and important ones are selected as the attributes of our case study.

HEC MONTRÉAL

Task and Dataset

- Almost any data have missing or inconsistent values.
- Therefore, to get a better model, data cleansing is inevitable.
- Thus, we start our data preparation by filtering missing values and removing inconsistent data and missing data.

Task and Dataset

- To determine delay label: scheduled arrival time - actual arrival time of buses has been calculated.
- Also, we remove the outlier 1’506 data with delay time higher than 2 hours.

Time labelled devoted to each record as delayed, earliness, on time

If the bus arrives ±5 minutes , we labelled on-time.

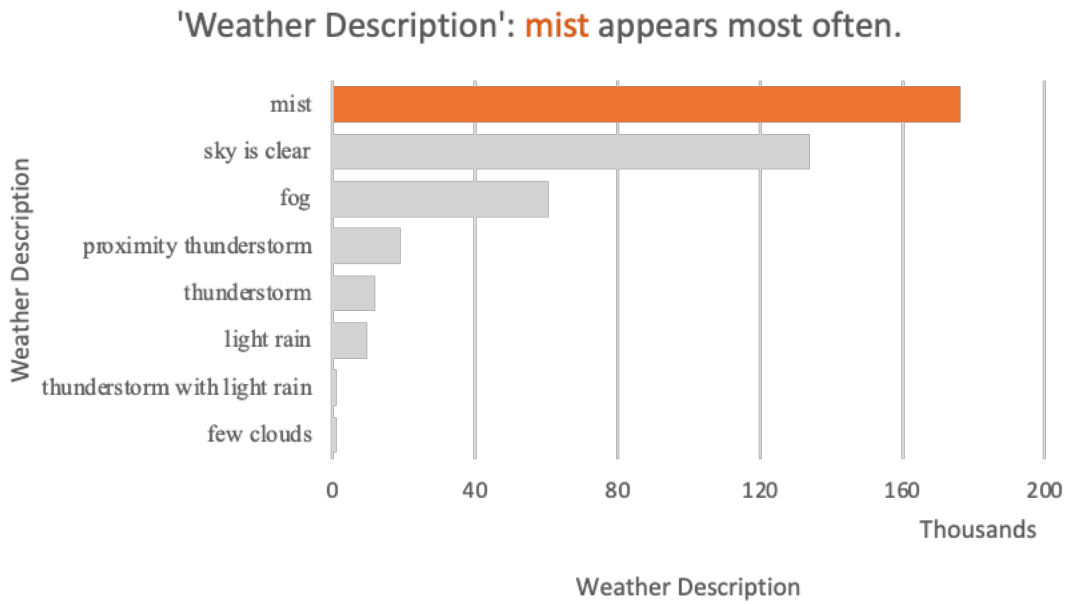
Between -40 minutes and -5 minutes we labeled that as earliness arrival.

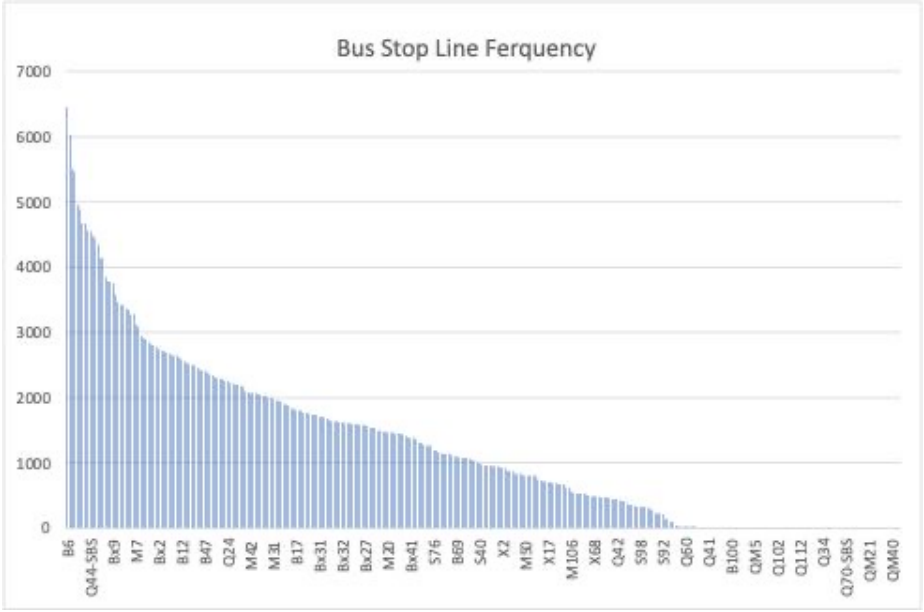
Between +5 minutes and +40 minutes , labeled that as a delayed arrival.

Task and Dataset

Statistics Related to data:

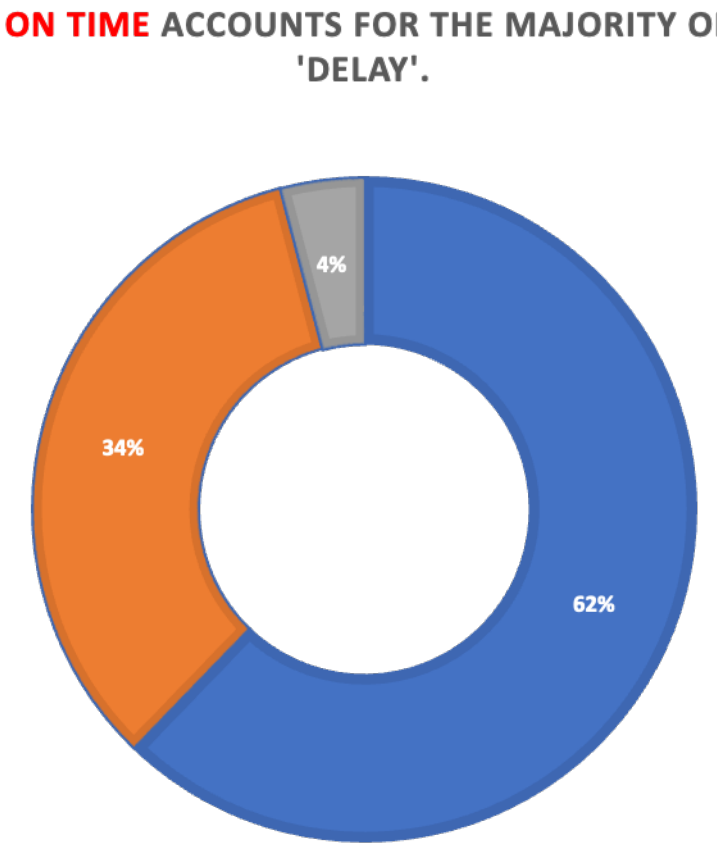
- The data records for the four weekdays of August of 2017.
- 330 different lines.
- 3824 stop points





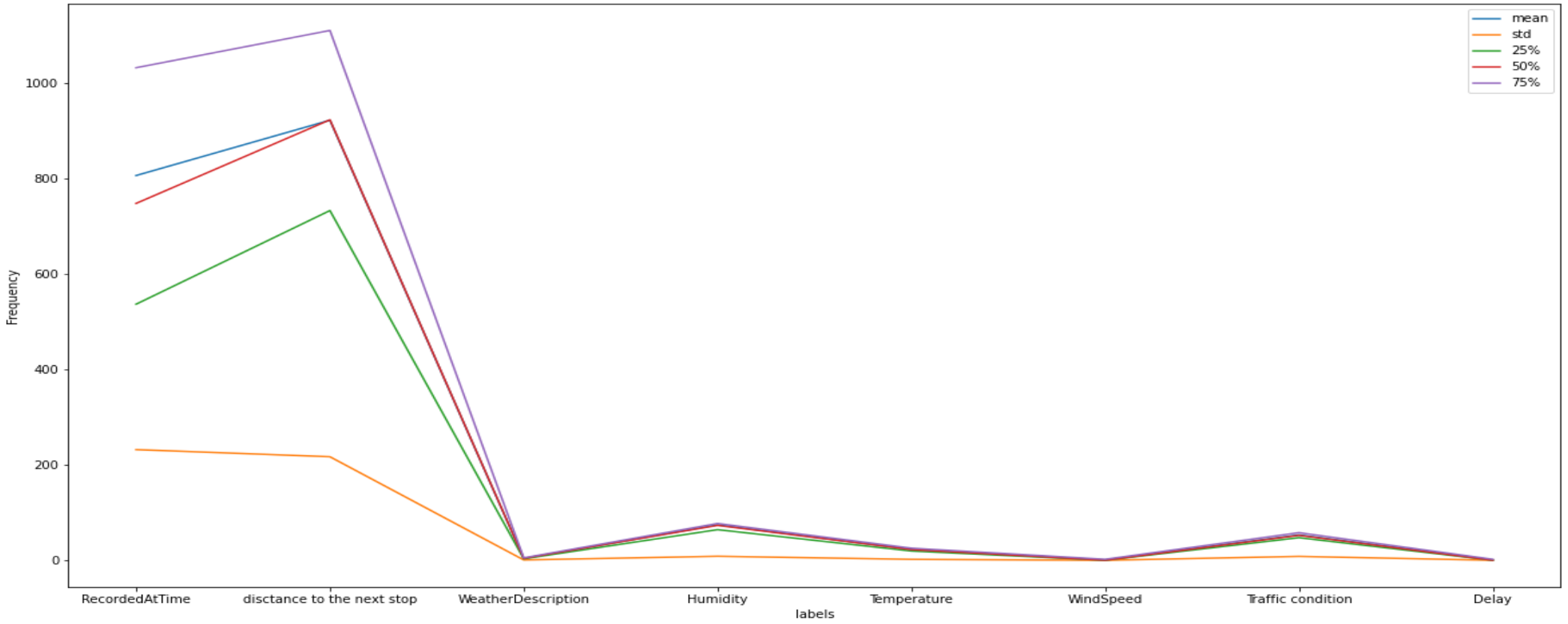
Task and Dataset

- Statics Related to data:



Labels	Count of Delay	
1	161,853	On Time
3	87,585	Delay
2	10,678	Earliness
Grand Total	**260,116**	

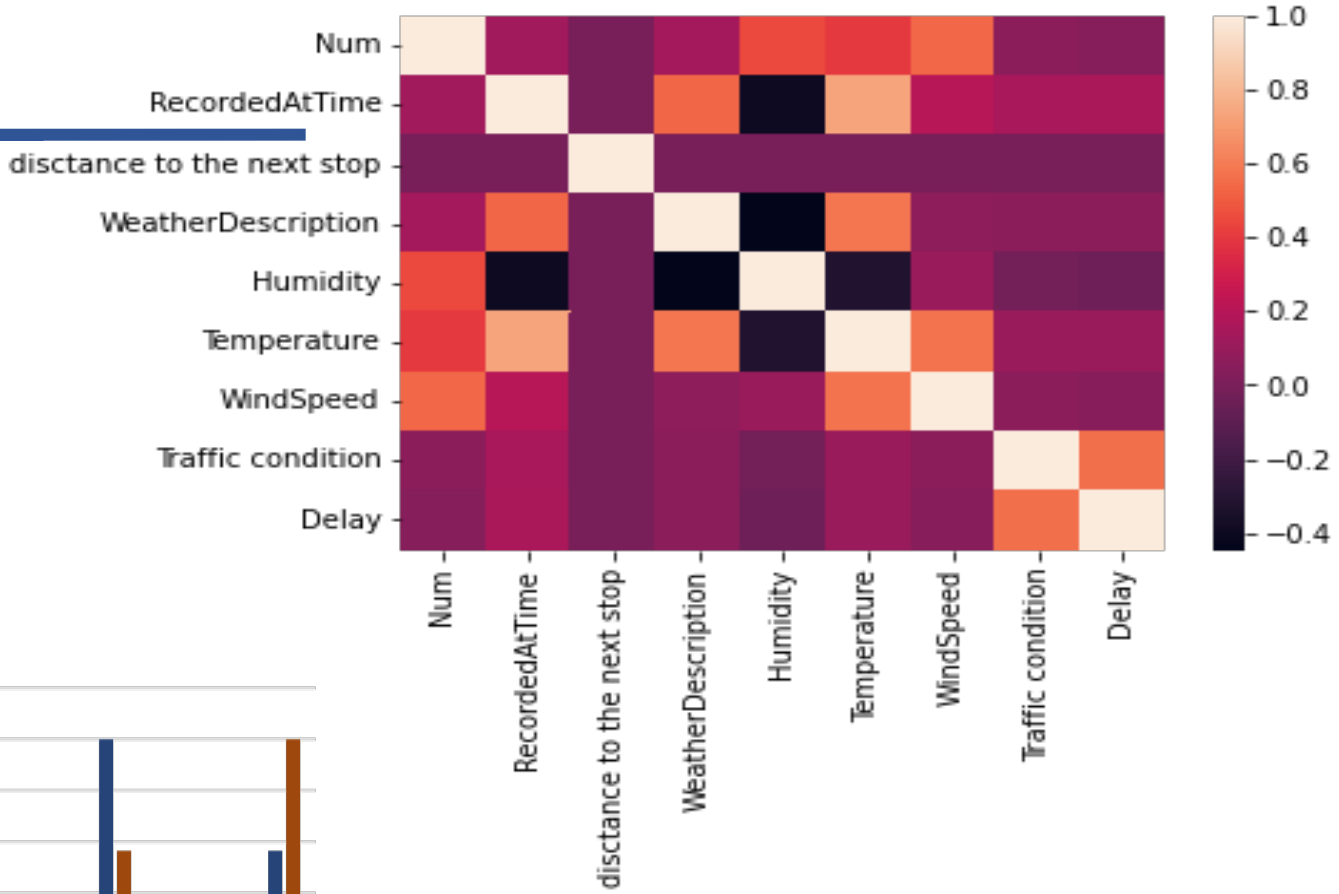
Task and Dataset

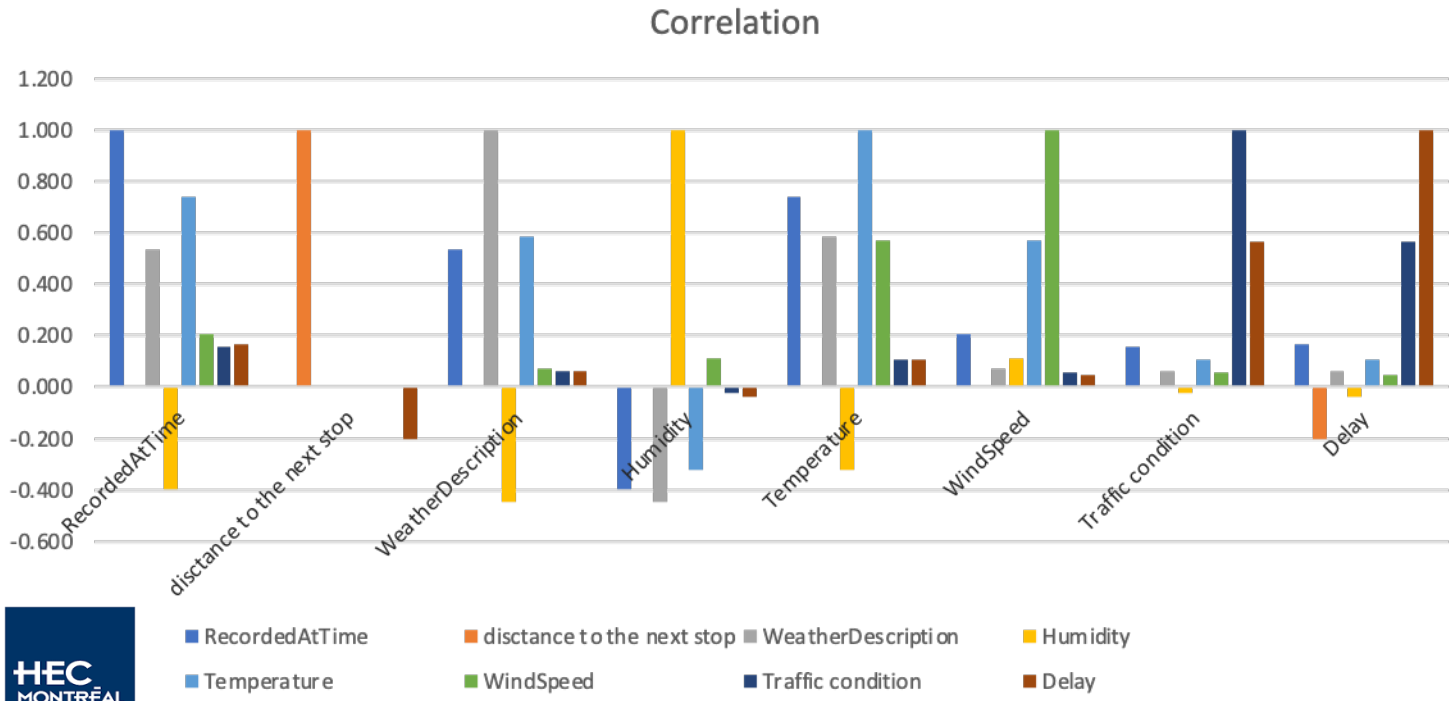


mean	806.3	922.1	4.6	75.1	23.7	1.7	54.1	1.7
std	232.6	217.7	1.6	9.3	2.9	0.9	9.1	0.9
min	445.0	545.0	1.0	54.0	19.8	1.0	1.3	1.0
50%	748.0	923.0	5.0	74.0	23.3	1.0	53.1	1.0
75%	1032.0	1110.0	6.0	78.0	26.3	3.0	58.7	3.0
Max	1093.0	1299.0	7.0	94.0	29.0	3.0	109.5	3.0

Task and Dataset

Correlation figure:





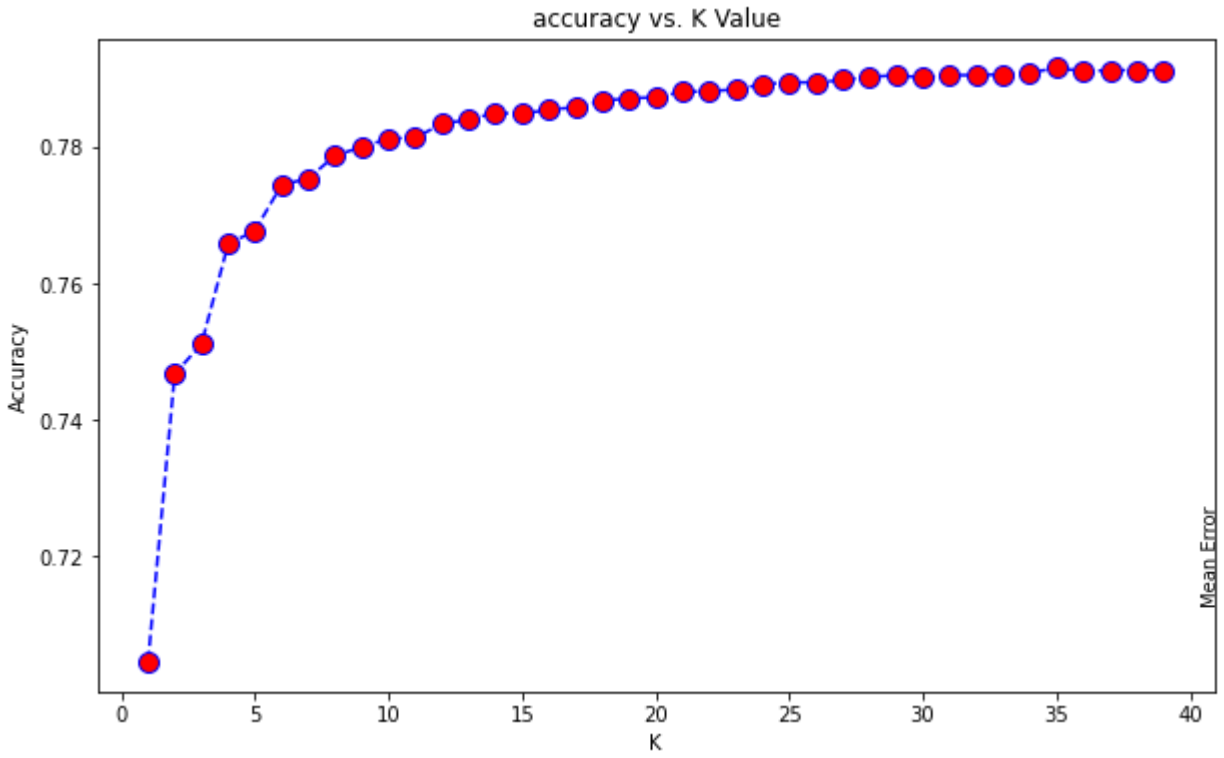
Literature review

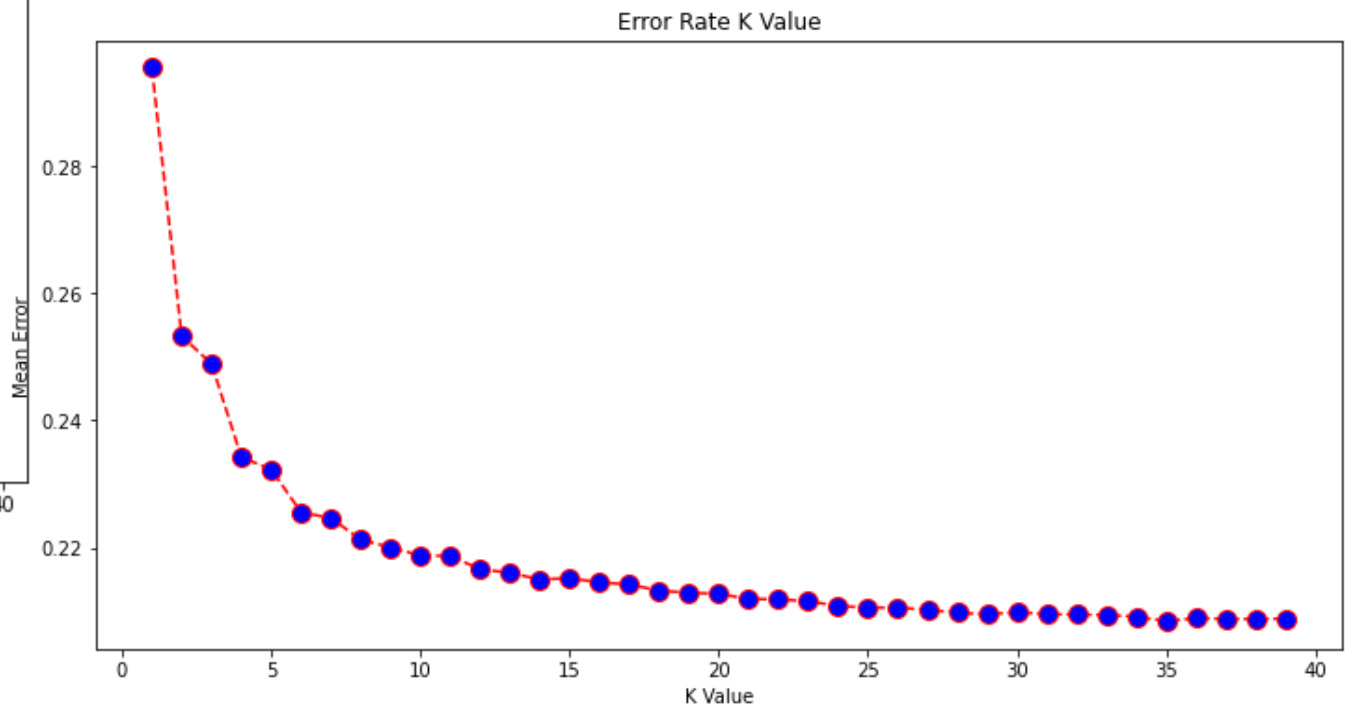
Author(s)	Objectives	Findings	Limitations	Method	Location
Liu et al. 2021	Bus travel-time prediction	Optimizing the Advanced Public Transportation System (APTS) effectiveness through improving the travel experience of bus riders	Focusing only on limited capacity of buses regarding bus delays and passengers flow	LSTM, **KNN**, fuzzy expert system	China
Kodiyan and Francis 2020	Predicting bus delays	Great accuracy of regression and ANN	Ignoring environmental aspects and accidents as predictors for delays	Regression, **DT,** ANN, and cloud ML	Dublin
Liu et al. 2020	Bus arrival time prediction	Proposing an efficient pre-processing approach for buses' GPS data for predicting bus arrival time.	Ignoring the seasonal variations in travel demand.	Regression, **KNN,** and neural network using LSTM	Dublin
Yamaguchi et al. 2018	Predicting bus travel and delay time	GBDT's better performance comparing to other adopted methods	Considering the data of different months of a year is neglected	ANN, RF, SVM, gradient boosting **decision tree** (GBDT), and linear regression	Fukuoka, Japan
Oruganti et al.2018	Predicting the travel time of bus transit	Understanding the effect of weather and traffic speed on travel time delays	Number of passengers during the day and distance of bus stops from downtown are overlooked	Linear Regression and **Random Forest**	Nashville
C. Coffey et al. 2011	Bus arrival time prediction	Finding an optimal number of neighbors k which changes as a function of the distance travelled	Only focused on one route	**k-Nearest Neighbors** (KNN)	Dublin

Solution Approach

K Nearest Neighborhoods

- Maximum accuracy: 0.79 at K = 34.





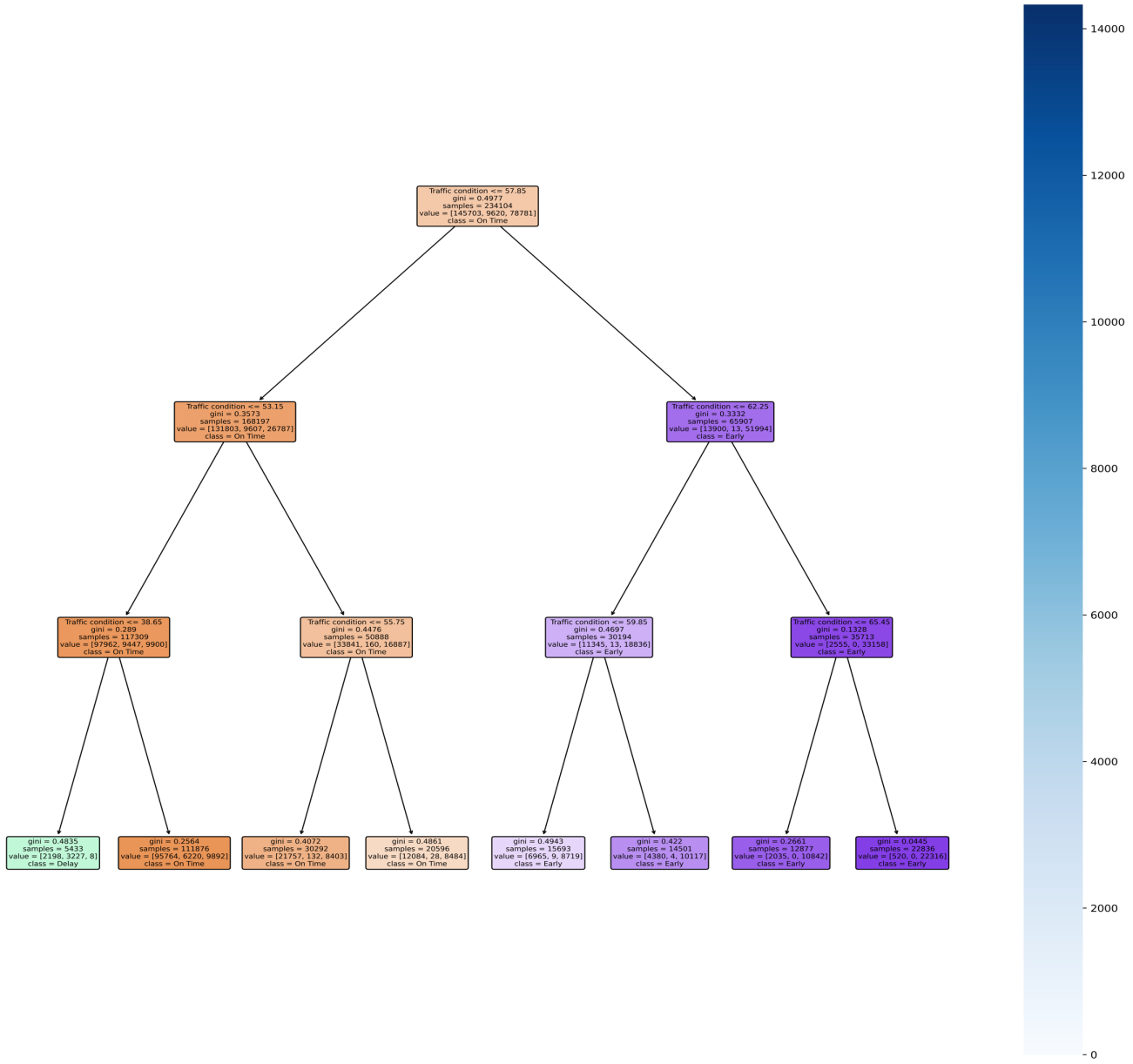
Solution Approach

Decision tree:

1) Tree depth
2) Sample spilt

Accuracy: 0.79

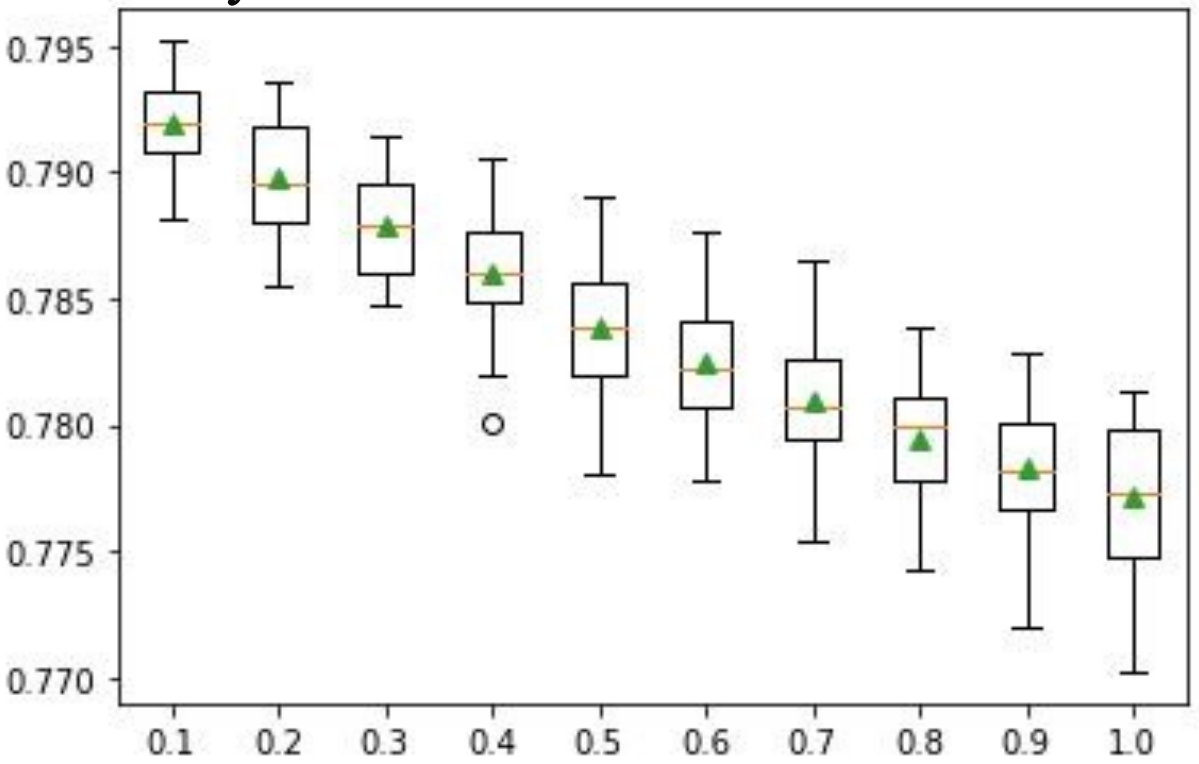
	precision	recall	f1-score	support
1	0.80	0.89	0.84	16150
2	0.56	0.34	0.42	1058
3	0.79	0.66	0.72	8804
accuracy			0.79	26012
macro avg	0.71	0.63	0.66	26012
weighted avg	0.78	0.79	0.78	26012

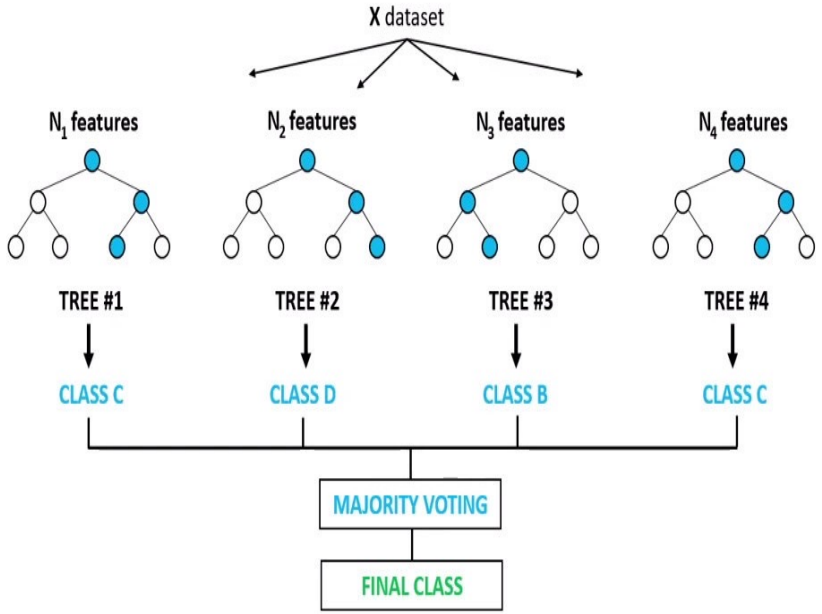


Solution Approach

Random Forest: Bootstrapping + Aggregation
1) Number of estimator 2) Number of splits

Accuracy: 0.792





Breiman, L. Random Forests. Machine Learning **45,** 5–32 (2001). https://doi.org/10.1023/A:1010933404324

Results

Algorithms	Accuracy	F1 score
Decision Tree	79 %	80 %
Random Forest	79.2%	84%
K Nearest Neighbors	79%	84%
Gradient boosting	65 %	66%

Future Direction

- To have a better prediction, multifactor-based methodologies, combining discrete and continuous analyses, can be proposed.
- Long-term period data can be collected.
- Other important features, such as Speed, Road type, Average travel time and bus incidents can be considered to improve model performance.
- Other ML algorithms, such as LSTM and XGBoost can be implied.
- Is these algorithms would yield similar results for different transport modes like Taxi?

HEC MONTRÉAL

Reference:

- X. Zhou, M. Wang, and D. Li, "Bike-sharing or taxi? Modeling the choices of travel mode in Chicago using machine learning," *Journal of transport geography,* vol. 79, p. 102479, 2019.
- A. Oruganti, F. Sun, H. Baroud, and A. Dubey, "Delayradar: A multivariate predictive model for transit systems," in *2016 IEEE International Conference on Big Data (Big Data)*, 2016, pp. 1799-1806: IEEE.
- S. Liu, X. Li, and C. He, "Study on dynamic influence of passenger flow on intelligent bus travel service model," *Transport,* vol. 36, no. 1, pp. 25-37, 2021.
- A. A. Kodiyan and K. Francis, "Prediction & Classification of Anomalies in Transport Network using Dublin Bus Transit Feeds."
- D. Liu, J. Sun, and S. Wang, "Bustime: Which is the right prediction model for my bus arrival time?," in *2020 5th IEEE International Conference on Big Data Analytics (ICBDA)*, 2020, pp. 180-185: IEEE.
- T. Yamaguchi, A. Mansur, and T. Mine, "Prediction of bus delay over intervals on various kinds of routes using bus probe data," in *2018 IEEE/ACM 5th International Conference on Big Data Computing Applications and Technologies (BDCAT)*, 2018, pp. 97-106: IEEE.
- S. I.-J. Chien, Y. Ding, and C. Wei, "Dynamic bus arrival time prediction with artificial neural networks," *Journal of transportation engineering,* vol. 128, no. 5, pp. 429-438, 2002.
- C. Coffey, A. Pozdnoukhov, and F. Calabrese, "Time of arrival predictability horizons for public bus routes," in *Proceedings of the 4th ACM SIGSPATIAL International Workshop on Computational Transportation Science*, 2011, pp. 1-5.
- Breiman, L. Random Forests. Machine Learning **45,** 5–32 (2001). https://doi.org/10.1023/A:1010933404324.


Thank You
Multumesc
Matondo
Terma Kasih
Obrigado
Merci
Danke
Diolch
Спасибо
ありがとう
Danke
Kiitos
Спасибо
Thank You
감사합니다
Teşekkür ederim
Shukran
Merci
Danke
Thank You
Obrigado
Merci
Welalin
Arigatô
Asante
ありがとう
Thank You
Спасибо
Danke
감사합니다
Dhanyavād
Obrigado
Dank u
Teşekkür ederim
Dank u
Shukran
Thank You
Diolch
Спасибо
Arigatô
Xièxiè
Thank You
Maake
ありがとう
Thank You

Team 2

Diabetes Prediction using Machine Learning

Presented by:
Oi Shan Pang
Jiaqi Yu

Introduction

According to CDC:

- More than 37 million people in the United States have diabetes, and 1 in 5 of them don't know they have it.
- 96 million US adults—over a third—have pre-diabetes, and more than 8 in 10 of them don't know they have it.
- Diabetes is the 7th leading cause of death in the United States (and may be underreported).

Task and Dataset

- Find the best machine learning model with highest accuracy for each dataset
- Using both Supervised & Unsupervised Models
- Find the most prominent features related to diabetes (e.g. BMI, Age)

Literature review

- A. Rajivkannan1, K. S. Aparna
  Survey On Diabetes Prediction Using Machine Learning Techniques
- M.S. Barale & D.T. Shirke
  Cascaded Modeling for PIMA Indian Diabetes Data
- Kayaer, Kamer, and Tulay Yldrm
  Medical diagnosis on Pima Indian diabetes using general regression neural networks
- Choubey, Dilip Kumar, et al.,
  GA_RBF NN: a classification system for diabetes on PIMA data
- Y. Angeline Christobel, P.Sivaprakasam
  A New Classwise k Nearest Neighbor (CKNN) Method for the Classification of Diabetes Dataset in PIMA
- Sara Belarouci , Mohammed Amine Chikh
  Medical imbalanced data classification

Dataset

Dataset - Vanderbilt	Dataset - Pima	Dataset - CDC
• Original data came from the Biostatistics program at Vanderbilt • Several hundred rural African-American patients were included. • 390 data point and 16 features	• Original data came from National Institute of Diabetes and Digestive and Kidney Diseases • -all patients here are females at least 21 years old of Pima Indian heritage. • 768 data point and 9 features	• Original data came from CDC • 400,000 Americans on health-related risk behaviors • Data for 2015 • 441,455 data point and 330 features

Dataset - Vanderbilt

	Patient number	Cholesterol	Glucose	HDL Chol	Chol/HDL ratio	Age	Gender	Height	Weight	BMI	Systolic BP	Diastolic BP	waist	hip	Waist/hip ratio	Diabetes
0	1	193	77	49	3.9	19	female	61	119	22.5	118	70	32	38	0.84	No diabetes
1	2	146	79	41	3.6	19	female	60	135	26.4	108	58	33	40	0.83	No diabetes
2	3	217	75	54	4.0	20	female	67	187	29.3	110	72	40	45	0.89	No diabetes
3	4	226	97	70	3.2	20	female	64	114	19.6	122	64	31	39	0.79	No diabetes
4	5	164	91	67	2.4	20	female	70	141	20.2	122	86	32	39	0.82	No diabetes

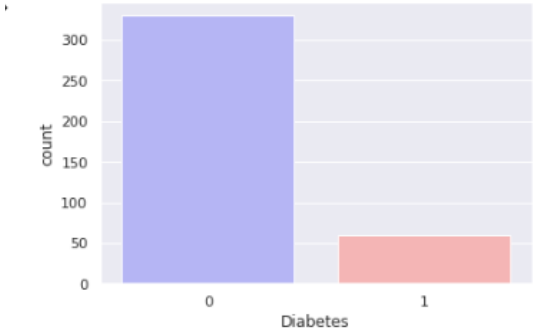
Dataset - Pima

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
3	78	50	32	88	31.0	0.248	26	1
2	197	70	45	543	30.5	0.158	53	1
1	189	60	23	846	30.1	0.398	59	1

Dataset - CDC

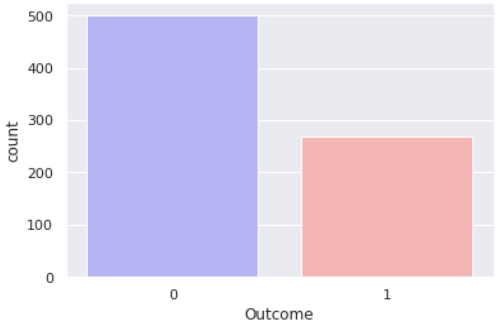
	Diabetes_binary	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	HvyAlcoholConsump	Sex	Age
0	0	1	1	1	40	1	0	0	0	0	0	9
1	0	0	0	0	25	1	0	0	1	0	0	7
2	0	1	1	1	28	0	0	0	0	0	0	9
3	0	1	0	1	27	0	0	0	1	0	0	11
4	0	1	1	1	24	0	0	0	1	0	0	11

Dataset - *Vanderbilt*



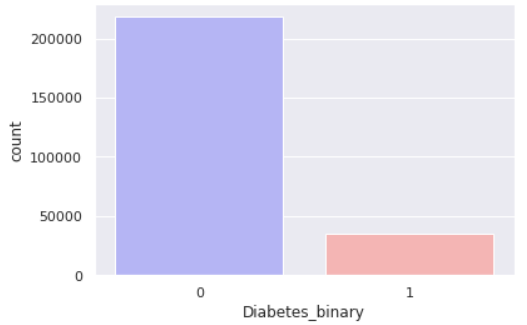
proportion of Diabetes patient 0.15384615384615385

Dataset - *Pima*



proportion of Diabetes patient 0.3489583333333333

Dataset - *CDC*



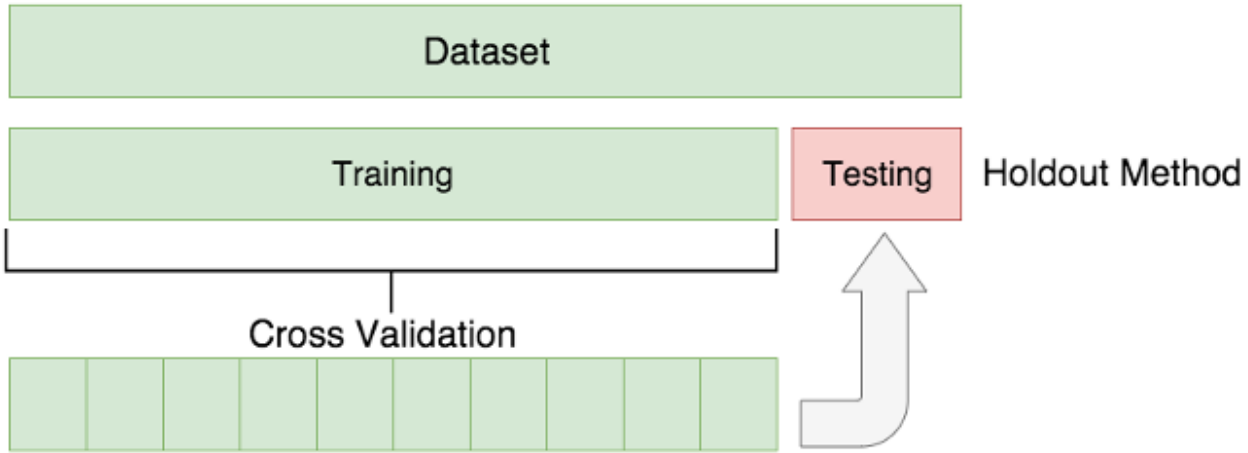
proportion of Diabetes patient 0.13933301797540207

Baseline Model – Majority Voting

Dataset - Vanderbilt	0.85
Dataset - Pima	0.65
Dataset - CDC	0.86

All predictions 0 – no diabetes

Data Splitting



Train: X,y
Test: X_holdout,y_holdout

Cross Validation

using Stratified K-fold Cross validation

```
train,holdout = train_test_split(diabetes_vanderbilt, test_size=0.2,random_state=123)
train=train.reset_index(drop=True)
holdout=holdout.reset_index(drop=True)
y = train.Diabetes
X = train.drop(["Diabetes","Patient number","Gender"], axis=1)
y_holdout = holdout.Diabetes
X_holdout = holdout.drop(["Diabetes","Patient number","Gender"], axis=1)

skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=333)
target = train.loc[:,'Diabetes']
skf.get_n_splits(X, y)
fold_no = 1

for train_index, test_index in skf.split(X, y):
  train_fold = train.iloc[train_index,:]
  test_fold = train.iloc[test_index,:]
  print('Fold',str(fold_no),'Class Ratio:',sum(test_fold['Diabetes'])/len(test_fold['Diabetes']))
  fold_no += 1
```

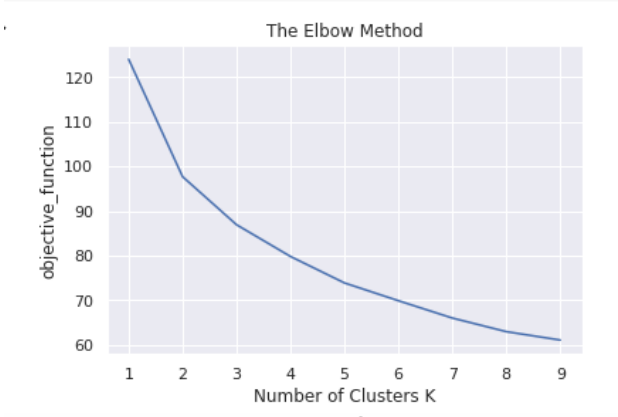
```
Fold 1 Class Ratio: 0.15873015873015872
Fold 2 Class Ratio: 0.14285714285714285
Fold 3 Class Ratio: 0.14516129032258066
Fold 4 Class Ratio: 0.14516129032258066
Fold 5 Class Ratio: 0.14516129032258066
```

Models

Unsupervised, otherwise supervised learning

- Logistic Regression
- Linear Discriminant Analysis
- KNeighborsClassifier
- Naïve Bayes
- Support Vector Machine
- Random Forest
- ADA Boosting
- Gradient Boosting
- Neural Networks MLPClassifier
- Kmeans Clustering

Hyperparameters



K means Clustering

Hyperparameters are tuned and selected for each model and dataset

```
models.append(('LR', LogisticRegression(max_iter=1000,C=0.01,penalty='l1',solver='liblinear',random_state=0)))
models.append(('LDA', LinearDiscriminantAnalysis(solver='svd')))
models.append(('NB', GaussianNB(var_smoothing=1e-11)))
models.append(('RF',RandomForestClassifier(bootstrap=True,max_depth=10,max_features='auto',min_samples_leaf=4,min_samples_split=5, n_estimators=10)))
models.append(('ADA',AdaBoostClassifier(n_estimators=50)))
models.append(('GB',GradientBoostingClassifier(learning_rate=0.1, max_depth=5, n_estimators=50)))
models.append(('MLP',MLPClassifier(activation='tanh',learning_rate_init=0.01,max_iter=4000)))
```

Accuracy

1

CDC

```
LR: 0.862080 (0.000684)
LDA: 0.860020 (0.000633)
KNN: 0.851171 (0.000766)
NB: 0.860893 (0.000010)
RF: 0.863145 (0.000736)
ADA: 0.863519 (0.000335)
GB: 0.864687 (0.000411)
```

Vanderbilt

```
LR: 0.923195 (0.025093)
LDA: 0.929544 (0.032619)
KNN: 0.932770 (0.032247)
NB: 0.897389 (0.021773)
SVM: 0.935996 (0.033152)
RF: 0.910292 (0.031018)
ADA: 0.932770 (0.033822)
GB: 0.929544 (0.032619)
MLP: 0.939222 (0.035234)
Kmean: 0.61
```

PIMA

```
LR: 0.771891 (0.030000)
LDA: 0.763748 (0.034146)
KNN: 0.728015 (0.027938)
NB: 0.749154 (0.030431)
SVM: 0.760536 (0.030247)
RF: 0.755671 (0.018217)
ADA: 0.747554 (0.013629)
GB: 0.771985 (0.017070)
MLP: 0.734466 (0.024102)
Kmean: 0.48
```

Important Features

CDC

	Gradient Boosting	Ada Boosting	Random Forest
0	HighBP	BMI	BMI
1	BMI	Age	HighBP
2	HighChol	HighBP	HighChol
3	Age	HighChol	Age
4	HeartDiseaseorAttack	Stroke	HeartDiseaseorAttack

Vanderbilt

	Support Vector Machine	Ada Boosting	Linear Disriminant Analysis
0	Glucose	Glucose	Gender_0
1	Age	Cholesterol	waist
2	Gender_0	HDL Chol	Height
3	Diastolic BP	Chol/HDL ratio	BMI
4	Height	Age	Glucose

PIMA

	Gradient Boosting	Linear Regression	Linear Discriminant Analysis
0	Glucose	BMI	DiabetesPedigreeFunction
1	BMI	Pregnancies	Pregnancies
2	Age	Glucose	BMI
3	DiabetesPedigreeFunction	Age	Glucose
4	Insulin	DiabetesPedigreeFunction	Age

- Glucose
- BMI
- Age
- High Blood Pressure
- High Cholesterol
- Gender

*Only top 3 most accurate models with coefficients or feature importance are considered here

Features by Gender

Female

CDC

	Gradient Boosting	Ada Boosting	Random Forest
0	HighBP	Age	BMI
1	BMI	BMI	HighBP
2	Age	HighBP	Age
3	HeartDiseaseorAttack	HighChol	HeartDiseaseorAttack
4	HighChol	HeartDiseaseorAttack	HighChol

Vanderbilt

	Support Vector Machine	ADA Boosting	Gradient Boosting
0	Glucose	Glucose	Glucose
1	Age	Cholesterol	Chol/HDL ratio
2	Diastolic BP	HDL Chol	Cholesterol
3	Height	Chol/HDL ratio	HDL Chol
4	Weight	Age	Age

PIMA

	Gradient Boosting	Linear Regression	Linear Discriminant Analysis
0	Glucose	BMI	DiabetesPedigreeFunction
1	BMI	Pregnancies	Pregnancies
2	Age	Glucose	BMI
3	DiabetesPedigreeFunction	Age	Glucose
4	Insulin	DiabetesPedigreeFunction	Age

Male

CDC

	Gradient Boosting	Random Forest	Ada Boosting
0	HighBP	BMI	BMI
1	BMI	HighBP	Age
2	HighChol	HighChol	HighBP
3	Age	Age	HighChol
4	HeartDiseaseorAttack	HeartDiseaseorAttack	HeartDiseaseorAttack

Vanderbilt

	Logistic Regression	Linear Discriminant Analysis	Ada Boosting
0	Glucose	waist	Glucose
1	Age	Glucose	Cholesterol
2	Systolic BP	Age	HDL Chol
3	Weight	Cholesterol	Chol/HDL ratio
4	BMI	Weight	Age

PIMA is female only

HEC MONTRÉAL

Minority=0.5*Majority

Oversampling Accuracy

CDC

```
LR: 0.741221 (0.000597)
LDA: 0.742137 (0.001583)
KNN: 0.711862 (0.000731)
NB: 0.666668 (0.000007)
RF: 0.750398 (0.001098)
ADA: 0.746017 (0.000845)
GB: 0.748776 (0.000843)
```

Mostly Lower compared to original sample

Vanderbilt

```
LR: 0.867215 (0.018410)
LDA: 0.857215 (0.019893)
KNN: 0.939873 (0.018304)
NB: 0.857215 (0.025411)
RF: 0.939873 (0.014492)
ADA: 0.949842 (0.015939)
GB: 0.942373 (0.009908)
MLP: 0.894652 (0.023677)
```

Mostly Lower compared to original sample

PIMA

```
LR: 0.764723 (0.052400)
LDA: 0.755468 (0.051750)
KNN: 0.712165 (0.043230)
NB: 0.746166 (0.039082)
SVM: 0.750829 (0.049044)
RF: 0.749267 (0.019996)
ADA: 0.730662 (0.039557)
GB: 0.749267 (0.025773)
MLP: 0.743160 (0.054988)
```

Mostly Lower compared to original sample

Holdout Test

	Best Model	Train Accuracy (CV)	Test Accuracy
CDC	Gradient Boosting	0.86	0.86
Vanderbilt	Neural Networks MLPClassifier	0.94	0.86
PIMA	Gradient Boosting	0.77	0.73

CDC is a larger dataset

Future Direction

Because the features can be highly correlated, will use Ridge and Lasso

More Data Exploration

THANKS!

Any questions?

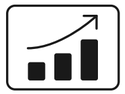
Team 3

TÉA GRAZZINI, JIALI LUO,
PIERRE ZWILLER PANICZ

Predicting stock price movement with Twitter data

Machine Learning Course

Problem definition



Stock markets are governed by random events and various sources of information can impact stock price fluctuation

Research shows that social media can influence the stock market

Twitter is an online platform where people share their emotions by tweeting about every topic

Task

Predict stock prices movement with Tweets expressing sentiments by using supervised learning models

Literature Review

Twitter mood predicts the stock market

- **Finding :**

Mood of tweet **can predict** the trend of the **Dow Jones Index**

- **Data date :**

Feb 28th to Dec 19th , 2008 **(10 months)**

- **Sentiment Analysis :**

(1) Opinion Finder and (2) GPOMS (6 moods)

Can Twitter Help Predict Firm-Level Earnings and Stock Returns?

- **Finding :**

Tweet Sentimet **can predict** firms' **quarterly** return and the abnormal stock returns around the earnings announcement

- **Data data :**

2006-2012 **(6 years)**

- **Sentiment Analysis :**

(1) Whole content sentiment and (2) word counts

Pipeline

01

Collect data

- Tweet dataset
- Finance dataset

02

Sentiment Analysis

- Extract Tweet sentiment
- VADER package

03

Merge datasets

- Merge by date
- One line per day & per company
- Compute financial metrics

04

Modelling

- Logistic Regression
- K-Nearest-Neighbor
- Multi Layer Perceptron
- Recurrent Neural Network

05

Peformance & Results

- Compare accuracy
- Plot results

01. Collect Data

Collect relevant data to perform our analysis

Tweets dataset about Tech companies

- From 12/07/2020 to 03/10/2020
- 1.133.004 observations
- 8 companies
- 15 variables

Stock prices dataset about Tech companies

- From 01/07/2020 to 31/10/2020
- 602 observations
- 8 companies
- 5 variables

Final dataset merging the tweets and the stock prices

- From 12/07/2020 to 31/10/2020
- 473 observations
- 8 companies
- 13 variables
  - Tweets + stock prices variables

02. Sentiment Analysis

Extract sentiment from Tweets using the VADER package

Lexicon based approach

- Uses a **pre-built dictionary** to find the sentiment score of words
- No need to train models using labeled data

Useful for social media text analysis

- Western style emojis
- Sentiment related acronyms (lol)
- Commonly used slang (nah & meh)

Step 1

Identify the Valence Score

- Matches score to a word
- Scales from -4 to 4 (most negative to most positive)
- 0 = neutral

Step 2

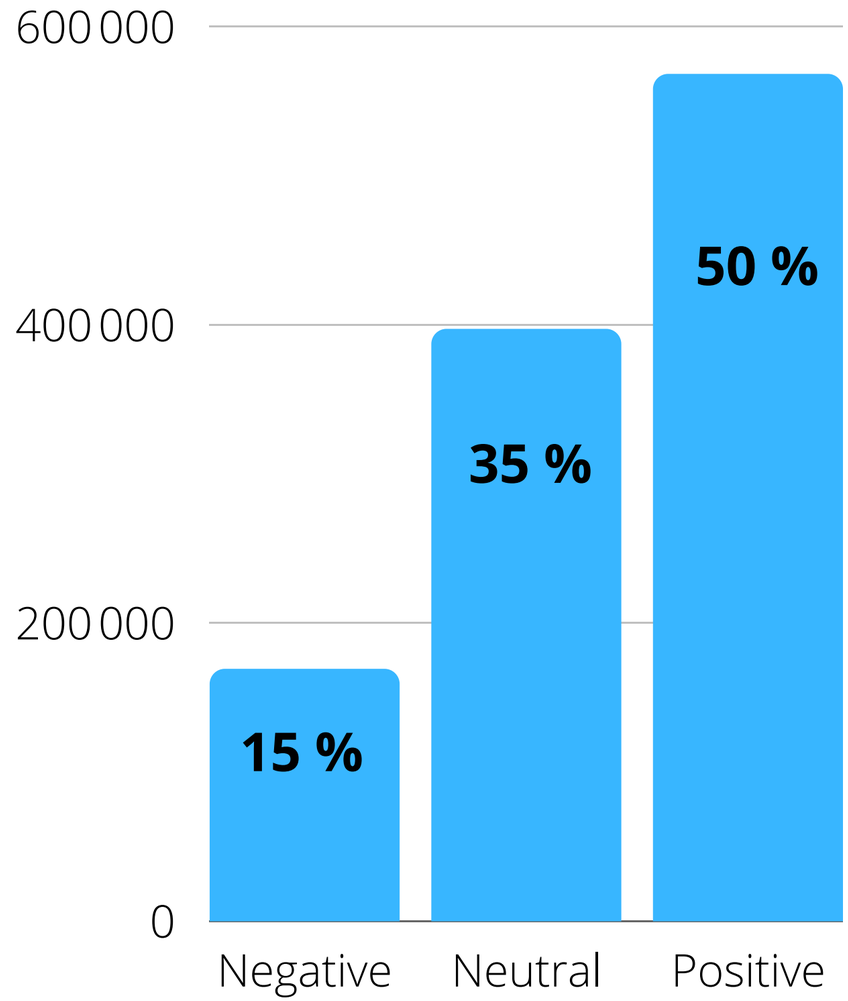
Compute the **Compound Score**

- Sum of valence scores of each word
- Normalize between -1 and 1 (most negative to most positive)

02. Sentiment Analysis

Sentiment proportion and Tweets examples

Tweet	Compound
Here's a list of 25 apps that Google removed from Play Store for stealing Facebook credentials	-0.31
What is The #Google Tag Manager? Learn Everything About GTM	0
I just got better PPC results thanks to seven Google Ads shortcuts #Google	0.83



03. Merge datasets

Aggregate the Twitter features by MEAN, MAX and MIN and by date and company

File Name	Date	Followers (min, max and mean)	Retweet Counts (min, max and mean)	Friends (min, max and mean)	Compound (min, max and mean)	Stock Price Movement
AMD	2020-07-08					-1

Stock price movement : **((Closing price – Opening price)/Opening Price) * 100** *(variation in %)*

The variation at day + 1 is used to determine the stock price movement :

- if value is > 0 then **stock price movement = 1**
- if value = 0: then **stock price movement = 0**
- if value < 0 then **stock price movement = -1**

04. Modelling

Split the dataset into train, validation and test sets

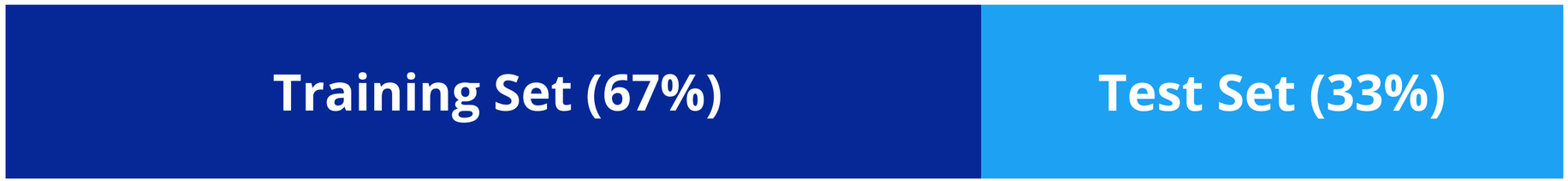
X

- **Friends** (min, max and mean)
- **Followers** (min, max and mean)
- **Retweet** (min, max and mean)
- **Compound** (min, max and mean)

Y

- **Stock price trend**

Categorical variable with 3 classes : positive (1), neutral (0) and negative (-1)



10-fold cross validation

04. Modelling

Training the models with 3 different supervised learning algorithms

Model	Reason
Logistic Regression	• Baseline model for binary classification
K-Nearest-Neighbor	• Low computation time • Similar sentiment, similar stock movement
Multi Layer Perceptron	• Good for nonlinear and complicated relationship

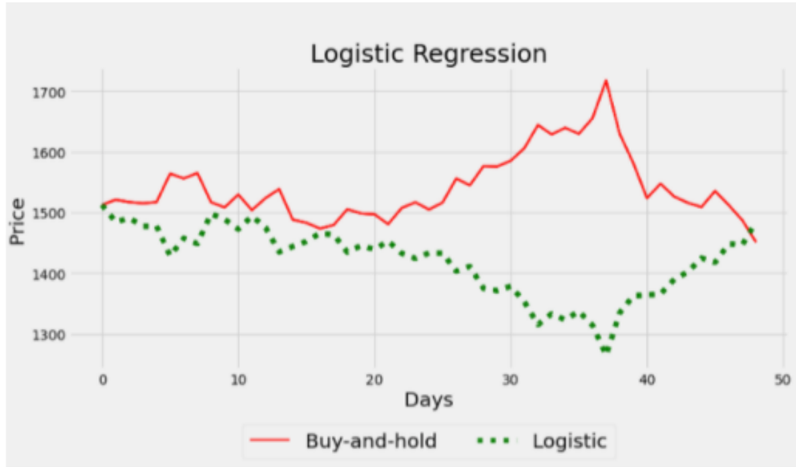
05. Performance

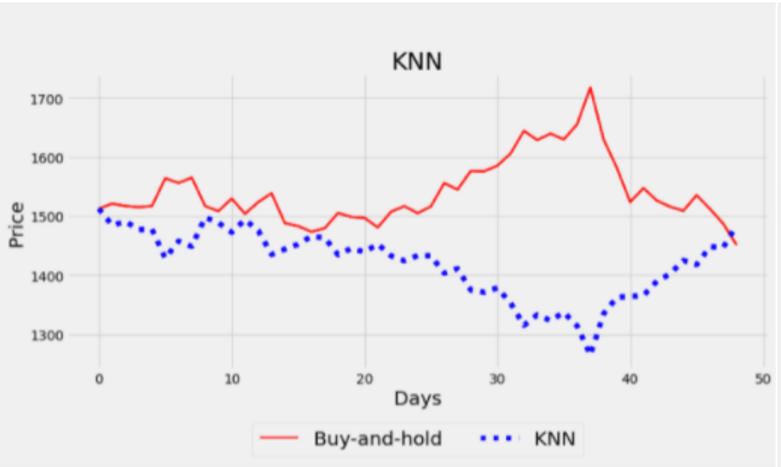
Model accuracy

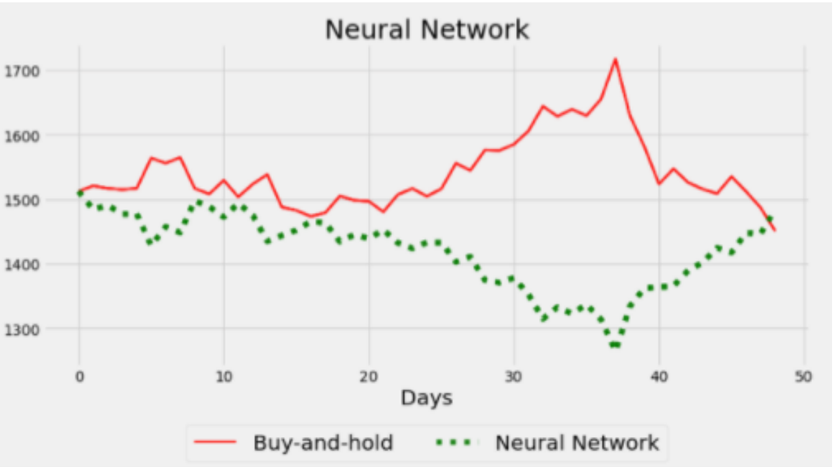
Model	Hyperparameters	Train Accuracy	Test Accuracy
Logistic Regression	• **Solvers : lbfgs** • **Penalty : 'l2'** • **C_values : 0.01**	**57.09%**	**53.98%**
K-Nearest-Neighbor	• Metric : Euclidean • N_neighbors : 2 • Weights : uniform	54.45%	47.78%
Multi Layer Perceptron	• Activation : tanh • Learning rate : 0.0001 • Constant • Hidden layersizes : (50, 50) • Solver : sgd	51.1%	43.3%

05. Results

Plots of the predicted movement of the stock price for Google

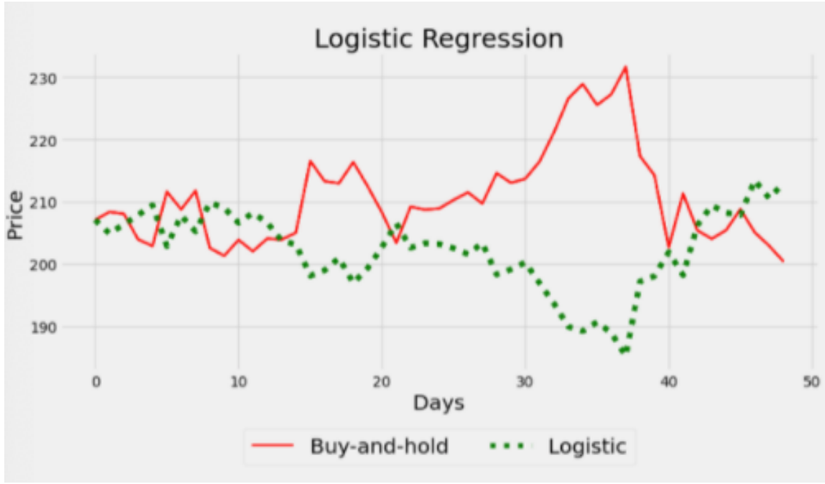


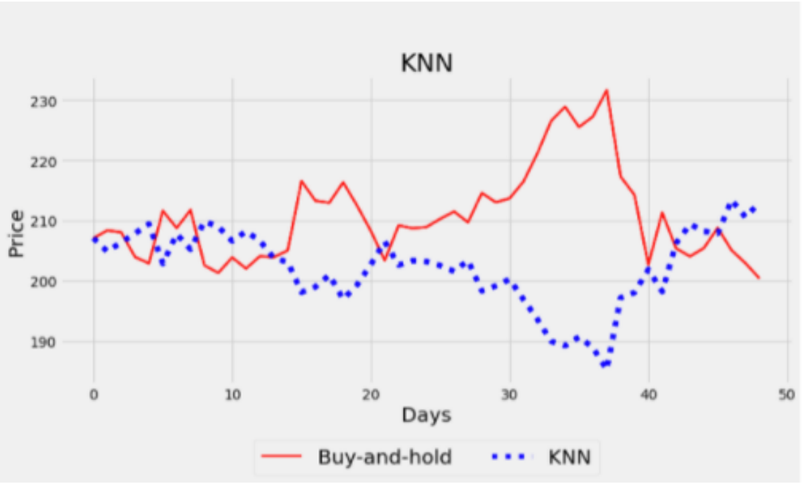


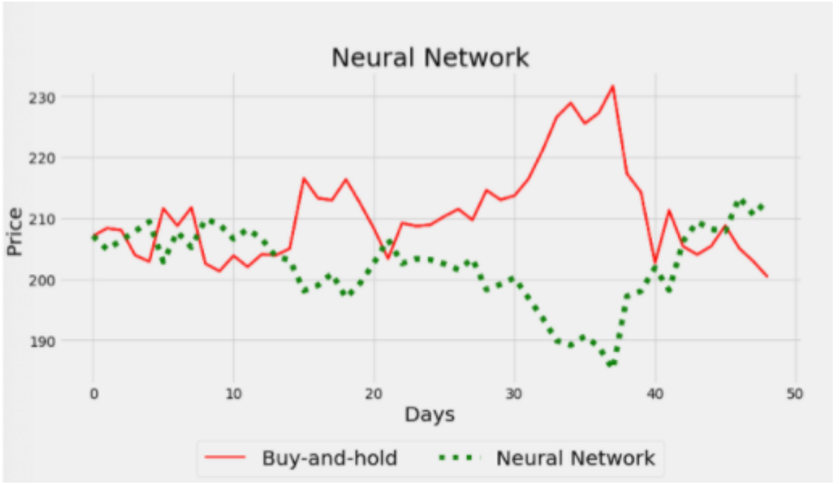


05. Results

Plots of the predicted movement of the stock price for Microsoft

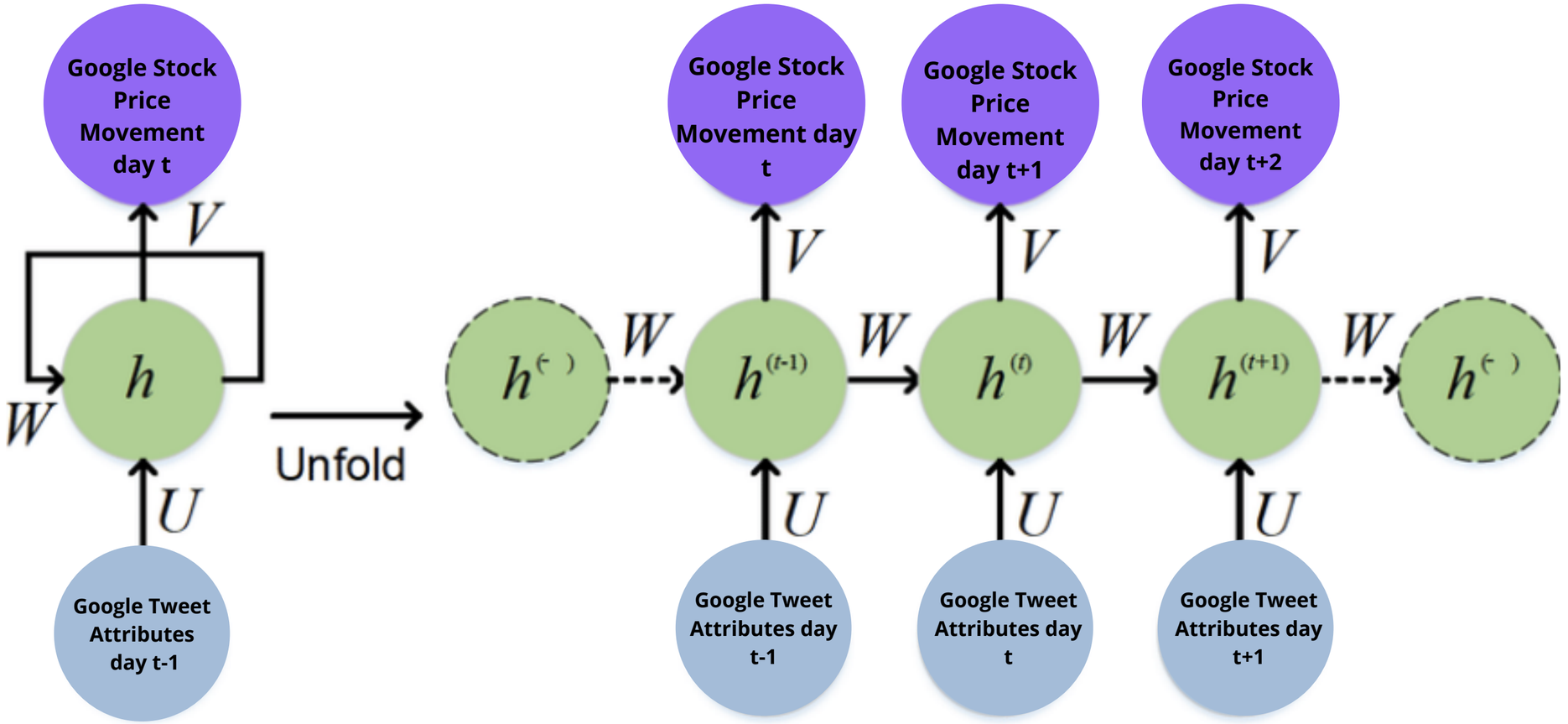




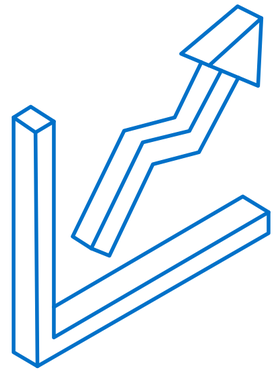


Method extension

Architecture of a Recurrent Neural Network for Google



Other Extension



Data

- Study analysts reports
- Use news information
- Use firms voluntary disclosure

Models

- Test other models
  - SVM
  - Naive Bayes
  - RNN

Output

- Abnormal returns
  - Can't be explained by financial information

Thank you!


18,089
7.08
4,310
53.21
8.26
0.11
0.04
3,300
2.02
0.41
1,313
13.13
1.00
0.09
2,788
0.44
0.13
14.41
3.91
0.40
1.94
1.01
-1.94
16.04
2.62
0.82
9.61
0.56
0.79
2.26
45.12
0.77
2.34
2.69
0.51
0.12

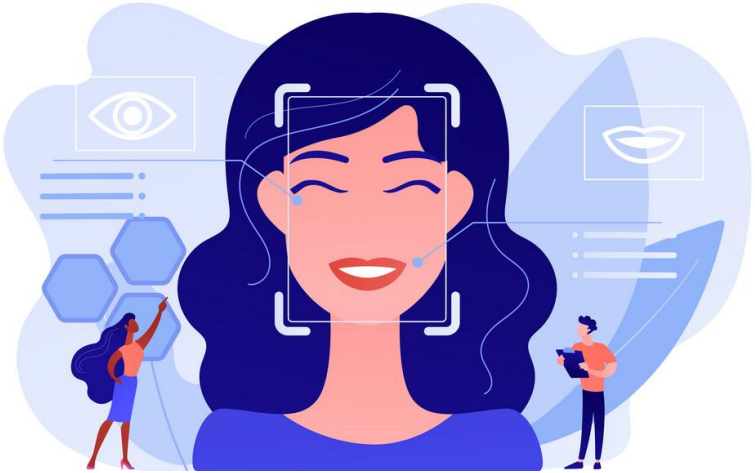
Team 4

HEC MONTRÉAL

Emotional recognition with machine learning

Christophe, Xi, Vincent

Task



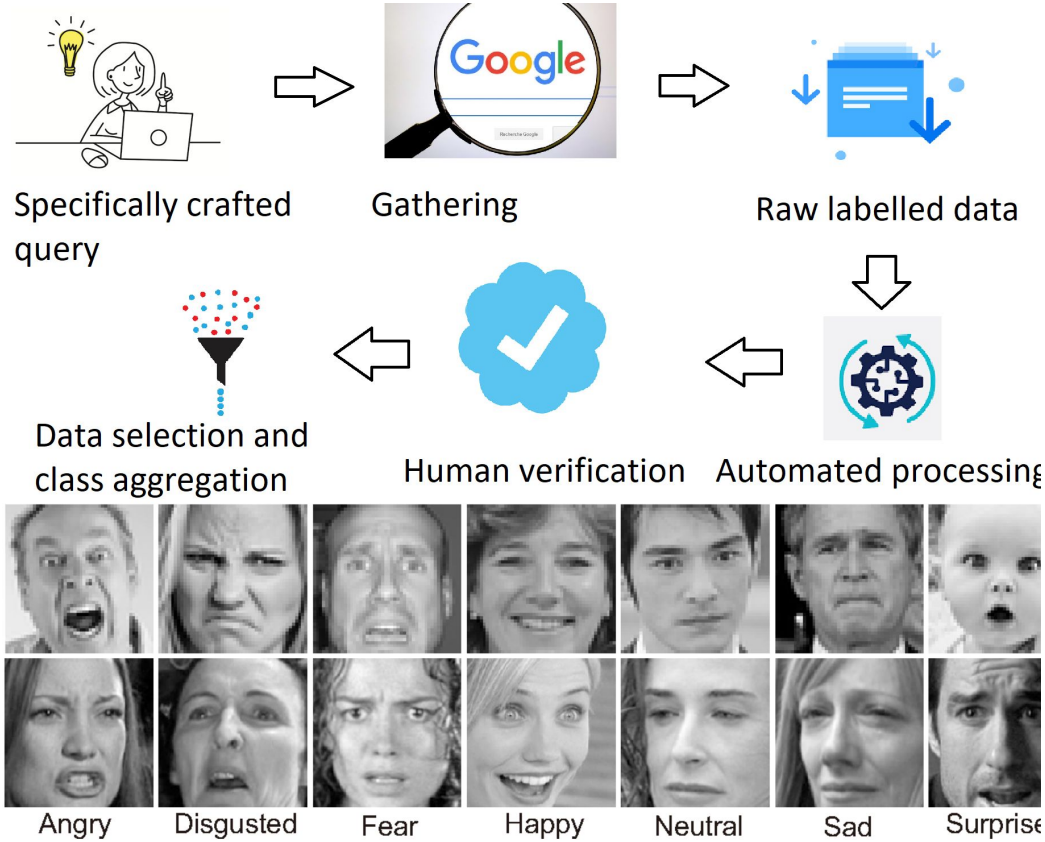
The task:

- Facial Emotional recognition (FER)
- Predict a label for an input image
- Input image preprocessed
- Supervised learning (Multi-class classification)

Dataset

A high quality & robust dataset:

- Crafted by professionals
- Lessen the bias
- Human labelling supervision
- Human testing of the dataset
- 35 887 observations



Literature review

Historic predecessors:

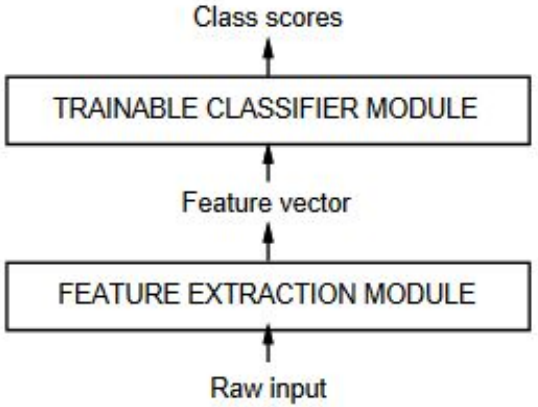
Hubel, D. H., & Wiesel, T. N. (1965)

- Not all neurons synapses fire when detecting retinal input
- Visual neurons are specifically wired
- Feature detecting structures “feed”
- Complex patterns emerge afterward and process these primary information
- 1 eyeball = 120M rods 6M cones
- Real time processing must be efficient

Literature review

Historic predecessors: LeNet-5

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998)



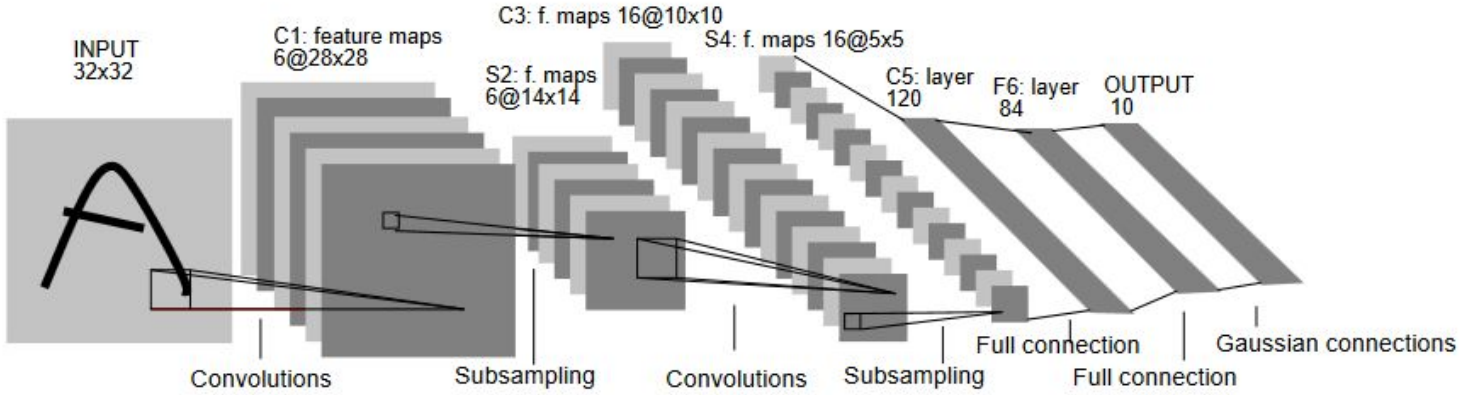
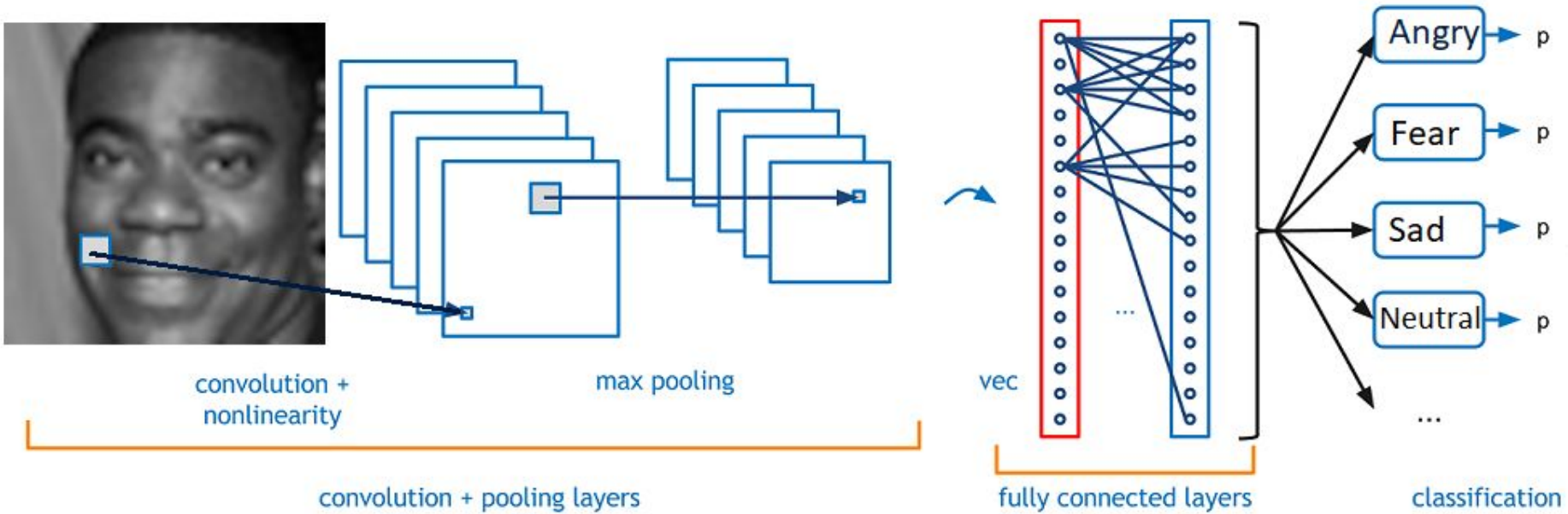


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Our approach based on the literature



A kernel exemple: the Sobel edge detector

3x3 part of the image

50	50	100
50	50	100
50	50	100

- 1		1
- 2		2
- 1		1

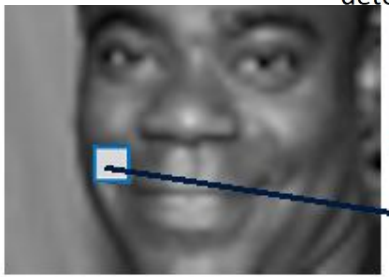
vertical edge detection kernel

=

- 50		100
- 100		200
- 50		-100

computation

= - 200 output



28 x 28

convolution + nonlinearity

26 x 26

max

22 x 22

-200	120	-40
0	20	10
25	-60	0

Maxpooling

= Max |value| =200

convolution + pooling layers

...

...ected layers

Angry p

Fear p

Sad p

Neutral p

...

classification

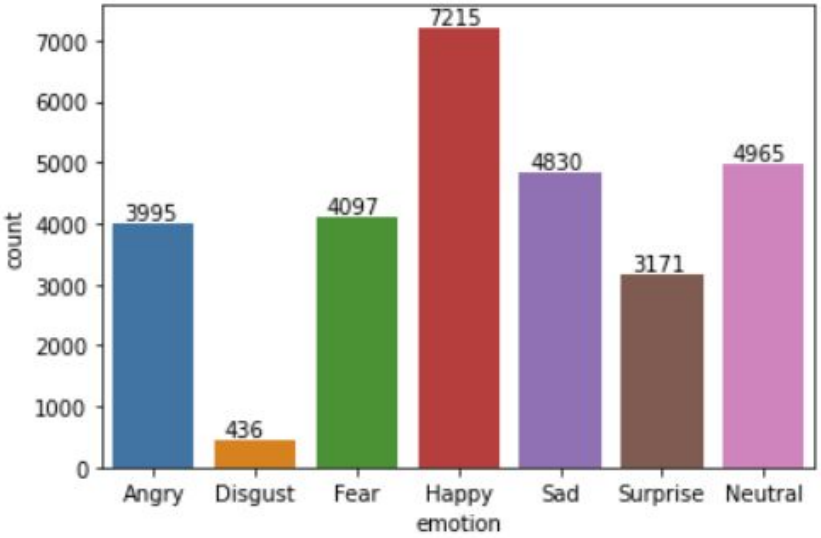
Evaluation Setup

Baseline model

Majority class

Max between-class imbalance level : 16,5x

Accuracy : 17.4% on test set

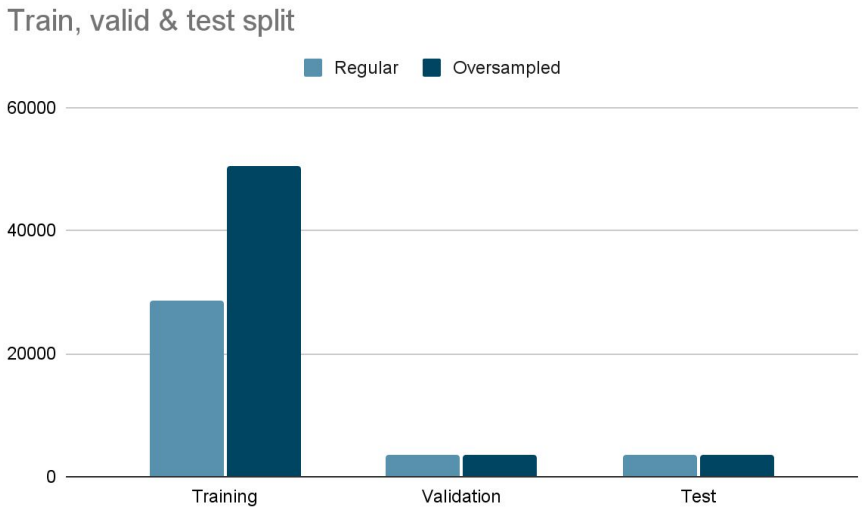


Evaluation Setup

Train, valid & test split

- According to the usage in the dataset :
  - 80% training
  - 10% validation
  - 10% test

- Regular split
- Oversampling



Evaluation Setup

Hyperparameters

01	Epochs	• 1 to 100
02	Learning rate	• 0.0001 to 0.01
03	Loss function	• Categorical cross-entropy • Categorical cross-entropy with custom regularizer
04	Layers configuration	• Filters size, count and stride • Number of layers • Pooling size

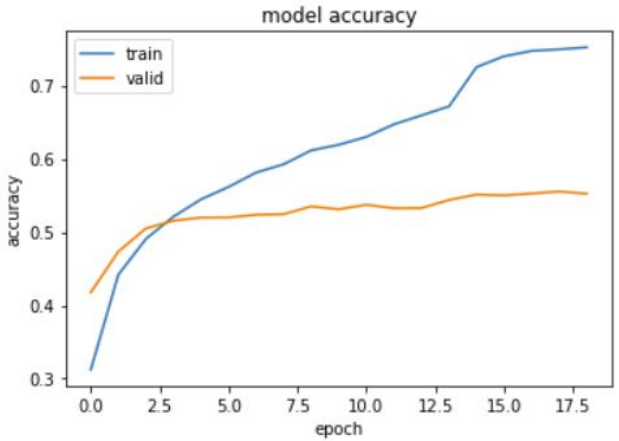
Evaluation Setup

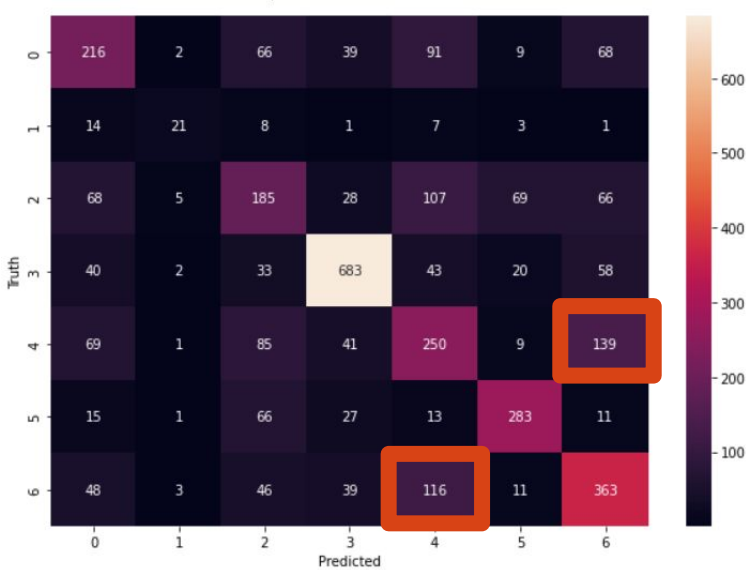
Hyperparameters

			Final model
01	Epochs	• 1 to 100	100 with Early Stopping
02	Learning rate	• 0.0001 to 0.01	0.001 with ReduceLROnPlateau
03	Loss function	• Categorical cross-entropy • Custom cross-entropy with regularizer	Categorical cross-entropy
04	Layers configuration	• Filters size, count and stride • Number of layers • Pooling size	ReLU and Softmax
			No oversampling

Results

Test accuracy : 55.8%





Sad

Neutral

Our attempt at regularizing

Categorical Cross-Entropy loss function :

$$\text{Loss} = -\sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i$$

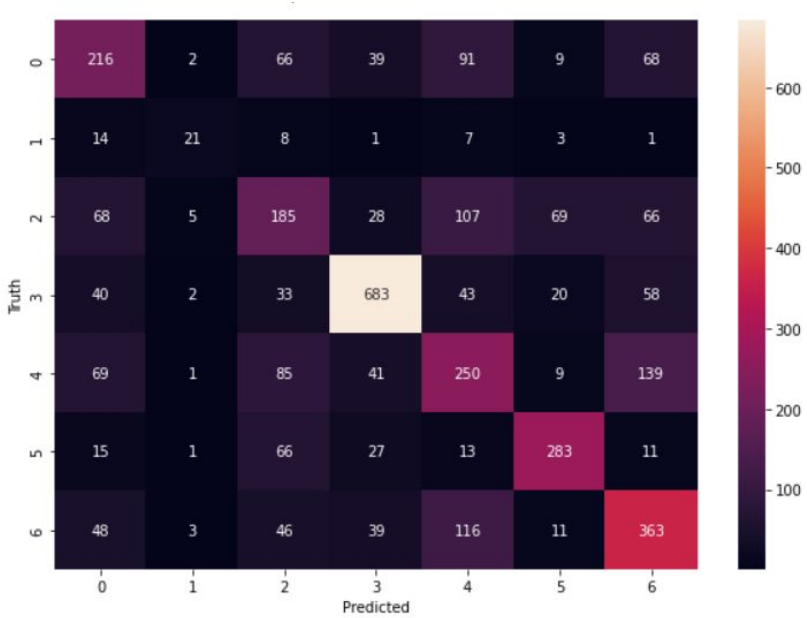
Custom regularizer used :

$$Regularizer = -k * \log(1 - \hat{y}_i)$$

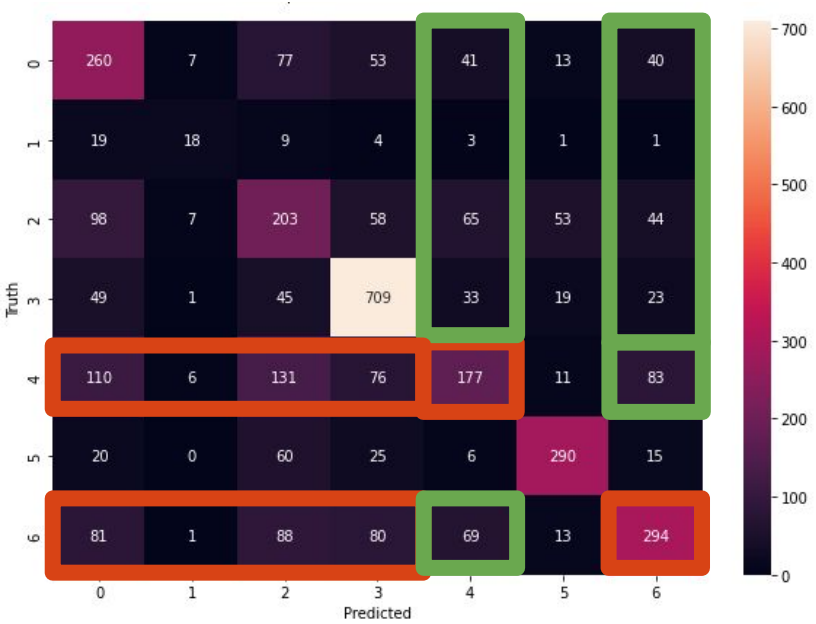
where $k = penaltyFactor$ and $i = \begin{cases} 4 \text{ if } y_{true} = 6 \\ 6 \text{ if } y_{true} = 4 \end{cases}$

Our attempt at regularizing

Regular Categorical Cross-Entropy (55,8% accuracy)



Custom loss function (54,4% accuracy)



Live test

HEC MONTRÉAL

Future Direction

- Better regularizer
- Ensemble models
  - Blending of models (best one in the competition was a blend of sparse filtering, random forests and SVM → 70,22% accuracy on test data)

Future Direction

Class imbalance problem

max between-class imbalance level of 16.5

- Data level
  - different data generation techniques
    - different resampling methods
    - data augmentation (top 3 teams did it in competition)
  - two-phase learning method

Future Direction

Class imbalance problem

- Algorithm level
  - performance metrics
    - F1-measure
    - a set of complementary performance metrics
  - loss functions
    - MFE (mean false error)
    - Focal loss
  - class weighting
    - increase the importance of minority class

HEC MONTRÉAL

Thank you!

Team 5

CREDIT CARD FRAUD

Chanel Larivière
Sonia Benghiat

01 OBJECTIVES

Objectives and data in hand

02 METHODOLOGY

Fixing the imbalance, finding the right model and tuning it

03 RESULTS ANALYSIS

Best model and prediction results for classification and LSTM

04 CONCLUSIONS

Difficulties, limits and possibles improvements

01.
OBJECTIVES

Objectives and data in hand

WHY STUDY CREDIT CARD FRAUD ?

- Multibillion-dollar “industry”
- It's expected to grow as people continue to rely on online shopping and social media.
- Increased by 44% between 2019 and 2020 according to the Federal Trade Commission (FTC).
- The trust in advancements of cyber-technology may become jeopardized if security of cardholders is sufficiently threatened.

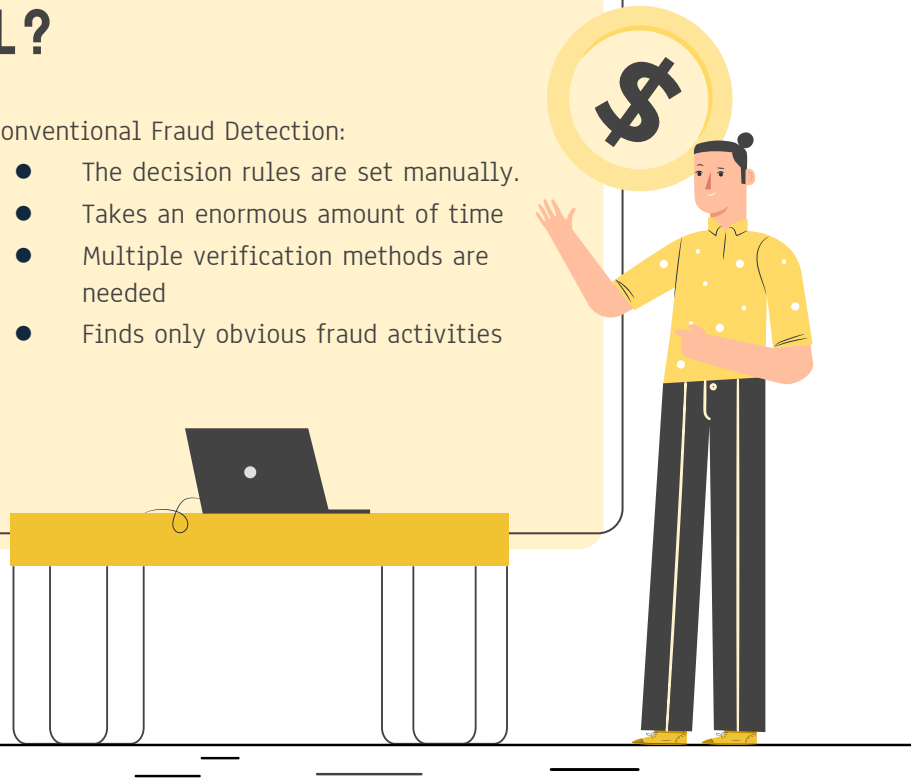
WHY ML ?

Machine Learning-based Fraud Detection:

- Detecting fraud automatically
- Real-time decision
- Less time needed for verification methods
- Identifying hidden correlations in data

Conventional Fraud Detection:

- The decision rules are set manually.
- Takes an enormous amount of time
- Multiple verification methods are needed
- Finds only obvious fraud activities



OBJECTIVES

T

TASK

Predict fraudulent transaction

P

PERFORMANCE

Confusion matrix (Sensitivity and AUC)

E

EXPERIENCE

Supervised learning

DATA

1 852 394

Data points

23

Features

FEATURES

NUMERIC VARIABLES	int	5 variables
DATES	datetime	2 variables
USER VARIABLES	nvarchar(50)	11 variables
MERCHANT VARIABLES	nvarchar(50)	4 variables

02.
METHODOLOGY

APPROACHES TRIED

Classification

LSTM

APPROACHES TRIED

Classification

LSTM

PIPELINE

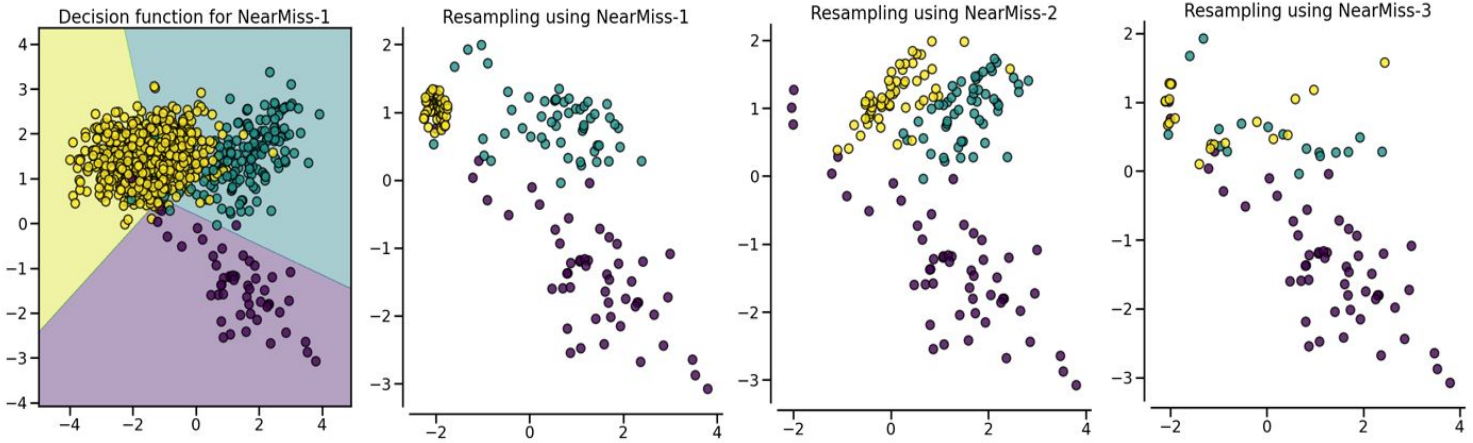
Data cleaning

Fixing the imbalance

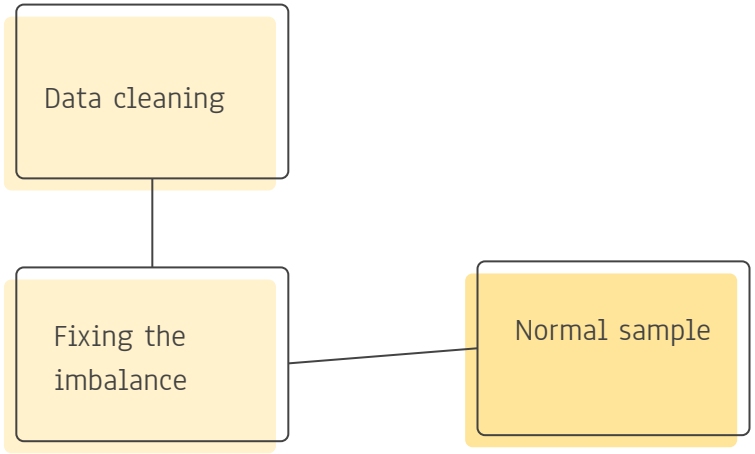
NearMiss3 undersample

Looks at the class distribution and randomly eliminates samples from the larger class.

1) Calculate the distance between the points in the majority class and the smaller class.
2) Select point based on distance with the smaller class.
3) Returns same amount of data as the minority class.



PIPELINE



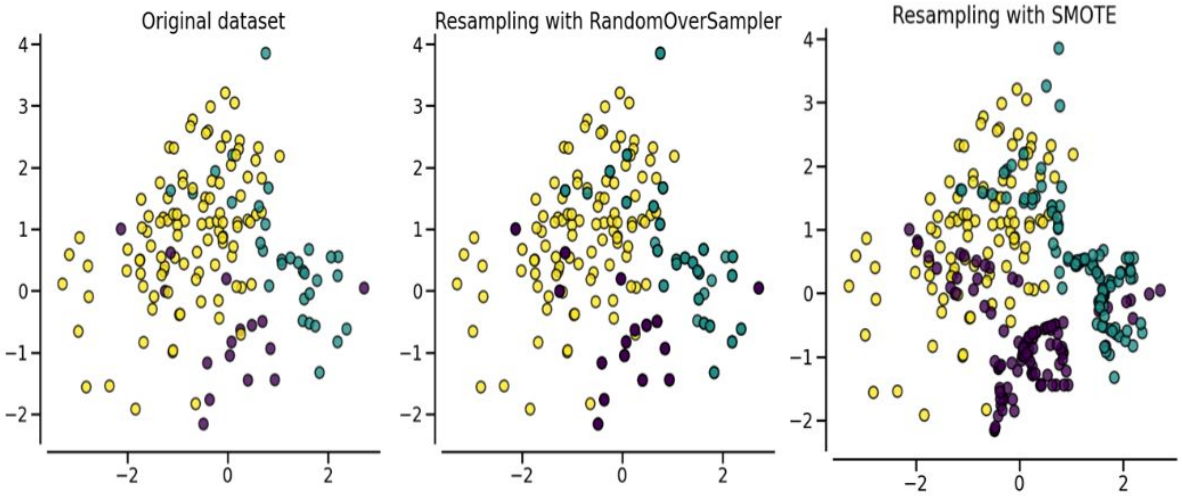
Using the class_weight = balanced attribute during modeling

PIPELINE

Data cleaning

Fixing the imbalance

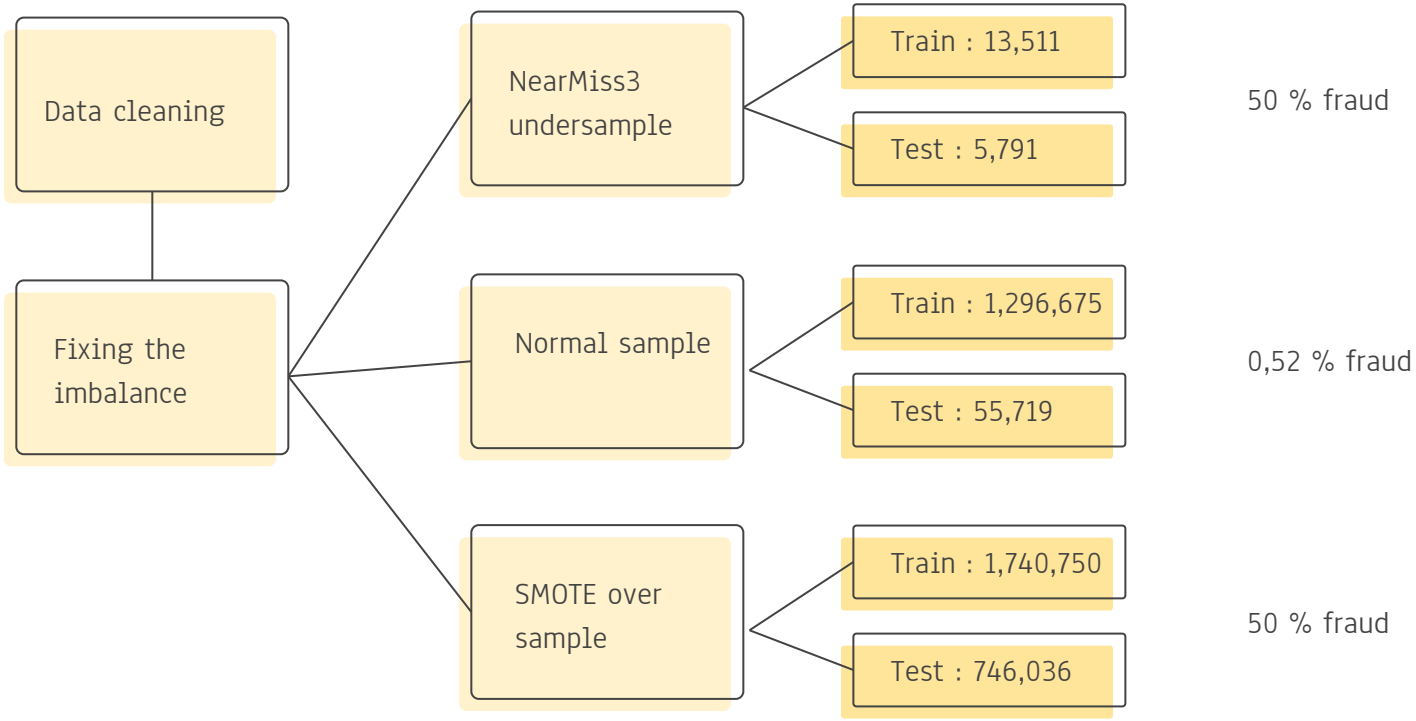
SMOTE over sample



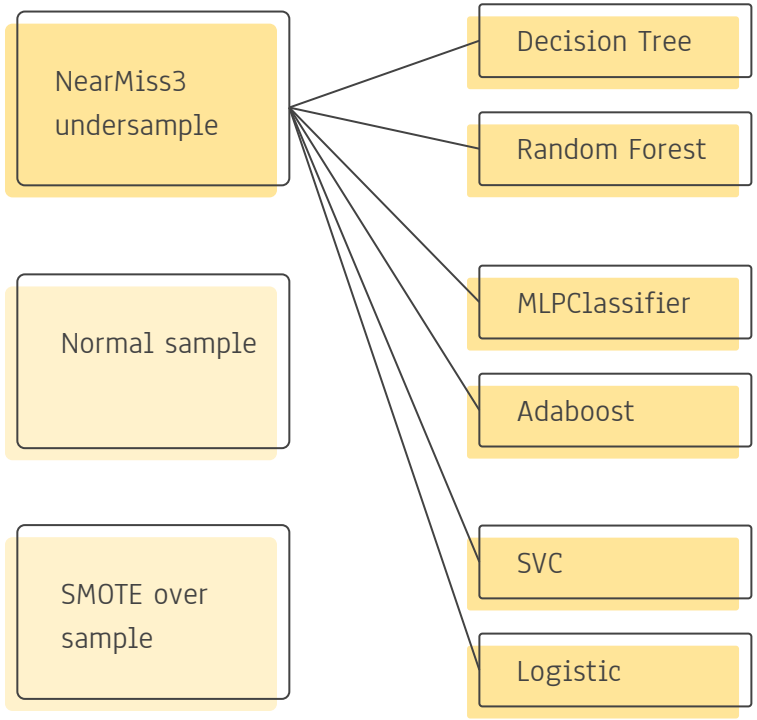
Synthesize new examples from the minority class

- Selects examples that are close in the feature space, drawing a line between the examples in the space and drawing a new sample at a point along that line.

PIPELINE


Data cleaning
Fixing the imbalance
NearMiss3 undersample
Train : 13,511
Test : 5,791
50 % fraud
Normal sample
Train : 1,296,675
Test : 55,719
0,52 % fraud
SMOTE over sample
Train : 1,740,750
Test : 746,036
50 % fraud

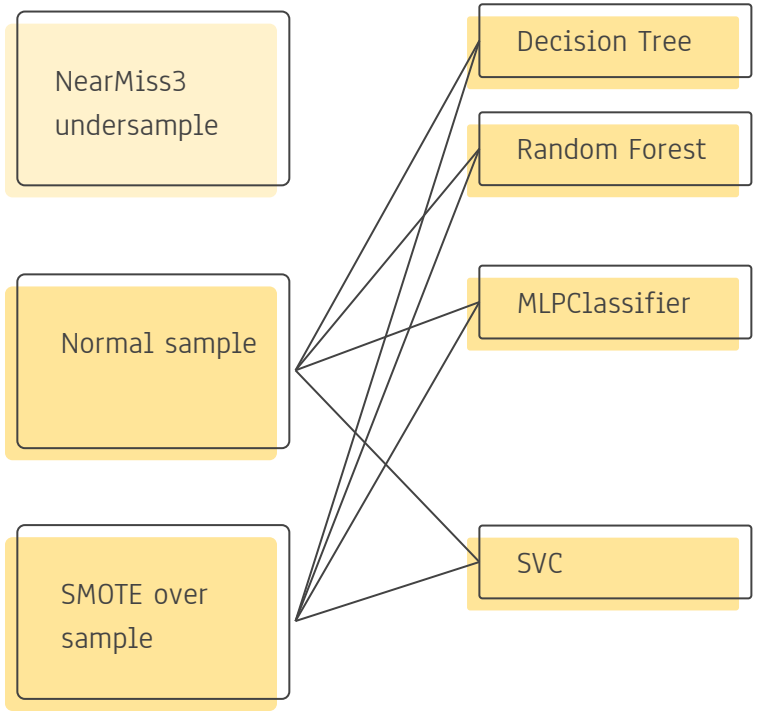
PIPELINE



Sensitivity : The proportion of actual fraud that are correctly identified.

	Model	Accuracy	Precision	Sensitivity	AUC
3	Decision Tree	0.8449	0.7911	0.8467	0.8449
1	Random Forest	0.8814	0.8361	0.8816	0.9532
	MLP	0.5517	0.5472	0.1150	0.6158
2	MLP - Standard	0.8518	0.7963	0.8664	0.9258
	Adaboost	0.7821	0.7229	0.7686	0.8738
	SVC	0.4985	0.4977	0.8087	0.5689
4	SVC - Standard	0.8225	0.7635	0.8315	0.8942
	Logistic	0.5000	0.5000	0	0.5677
	Logistic - Standard	0.7783	0.7164	0.7800	0.8439

PIPELINE



Decision Tree	Accuracy	Precision	Sensitivity	AUC
Under sample	0.8449	0.7911	0.8467	0.8449
Normal sample	0.9966	0.4687	0.6913	0.8448
Over sample	0.9965	0.9940	0.9981	0.9965

Random Forest	Accuracy	Precision	Sensitivity	AUC
Under sample	0.8814	0.8361	0.8816	0.9532
Normal sample	0.9978	0.5920	0.6406	0.9817
Over sample	0.9989	0.9984	0.9985	0.99996

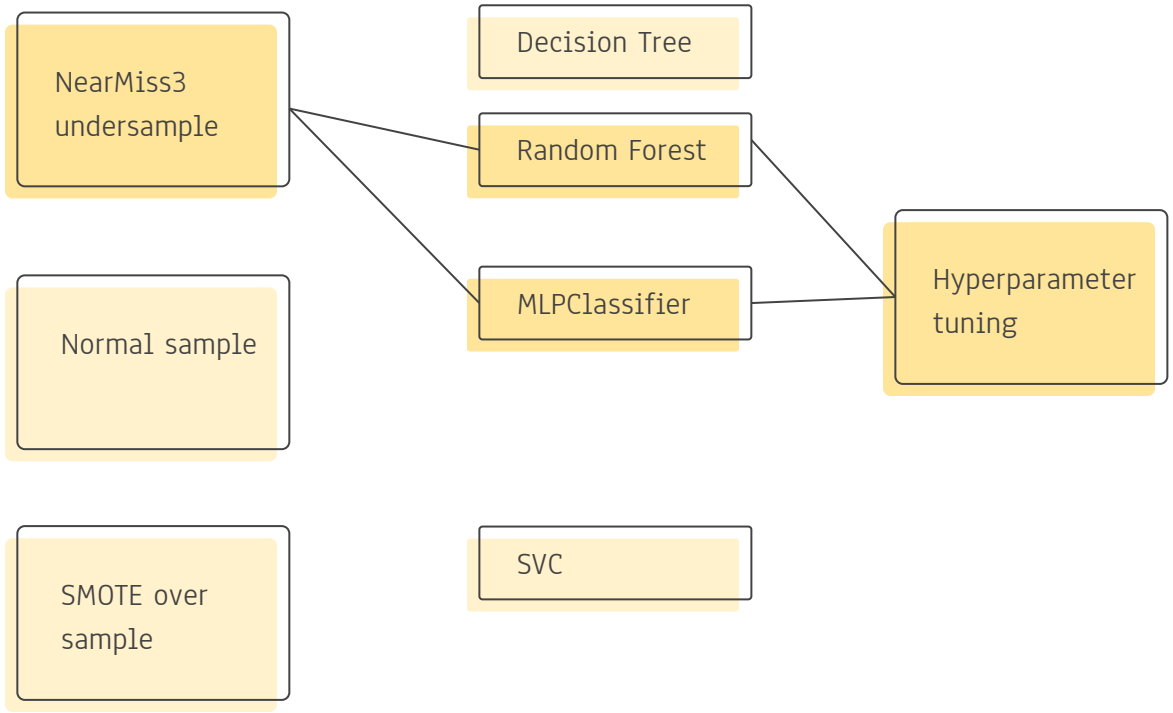
MLP - Standard	Accuracy	Precision	Sensitivity	AUC
Under sample	0.8518	0.7963	0.8664	0.9258
Normal sample	0.9978	0.5962	0.7067	0.4388
Over sample	0.9974	0.9956	0.9984	0.4906

SVC - Standardized	Accuracy	Precision	Sensitivity	AUC
Under sample	0.8225	0.7635	0.8315	0.8942
Normal sample				
Over sample				

Key takeaways :

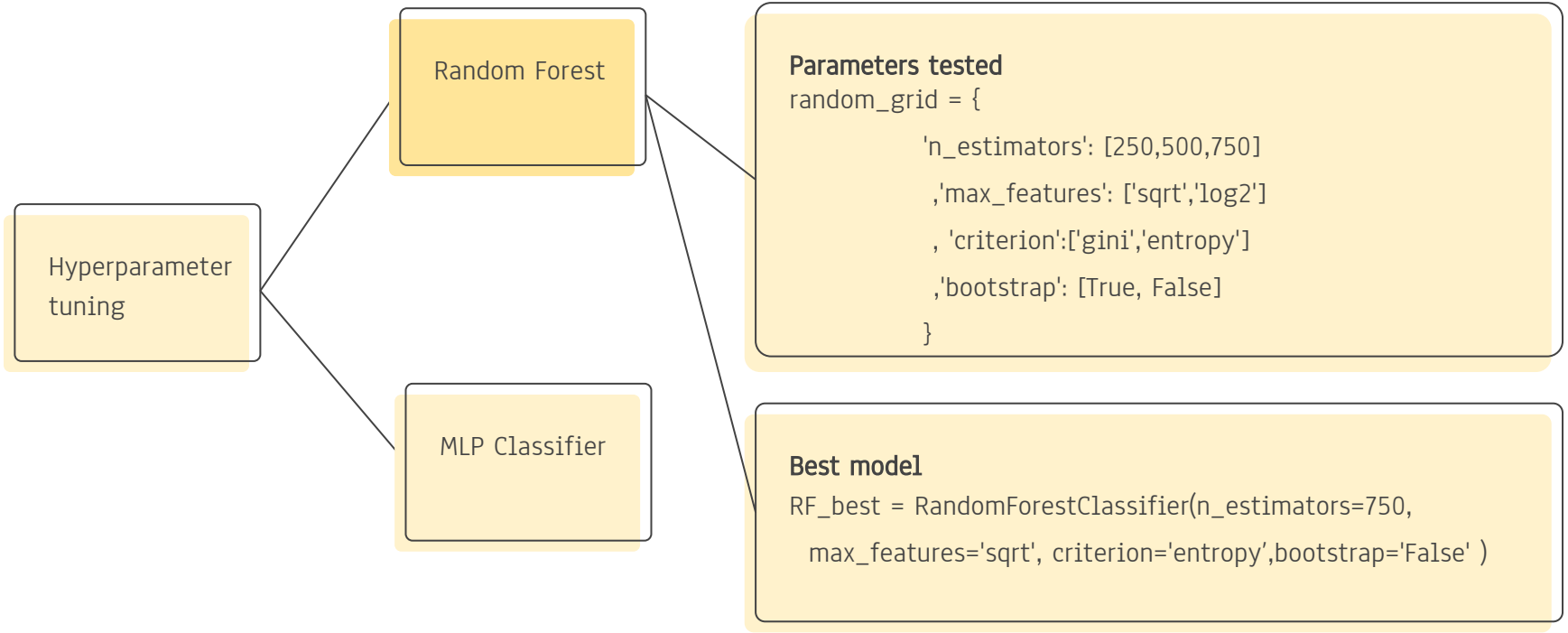
1) Prioritization on Sensitivity and AUC metrics as the focus is to measure how well a classifier can predict Fraud (TP).
2) The models that worked well on the under sample, work even better on the oversample.
3) Even when using the attribute 'class_weight=balanced' the normal sample wasn't able to correctly identify the fraud in the normal sample (lowest sensitivity).

PIPELINE



1) GridSearch
2) Cross validation

PIPELINE


Hyperparameter tuning
Random Forest
MLP Classifier
Parameters tested
random_grid = {
'n_estimators': [250,500,750]
,'max_features': ['sqrt','log2']
, 'criterion':['gini','entropy']
,'bootstrap': [True, False]
}
Best model
RF_best = RandomForestClassifier(n_estimators=750,
max_features='sqrt', criterion='entropy',bootstrap='False')

RESULTS

Hyperparameter tuning

Random Forest

Random Forest	Accuracy	Precision	Sensitivity	AUC
Under sample : baseline	0.8814	0.8361	0.8816	0.9532
Under sample : tuned	0.8860	0.8408	0.8902	0.9561

44%	6%
5%	45%

Random Forest	Accuracy	Precision	Sensitivity	AUC
Normal sample : baseline	0.9978	0.5920	0.6406	0.9817
Normal sample : tuned	0.9982	0.6601	0.7080	0.9960

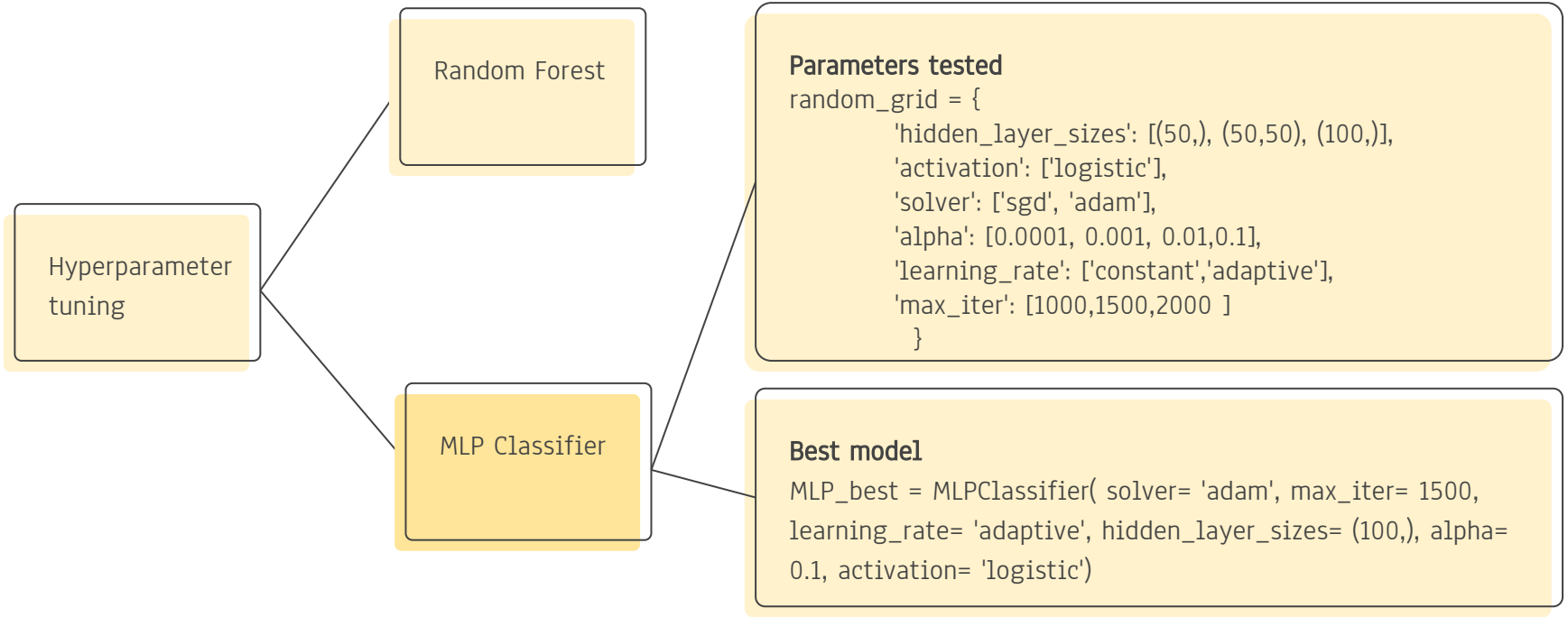
99,44%	0.03%
0.15%	0.37%

MLP Classifier

Random Forest	Accuracy	Precision	Sensitivity	AUC
Over sample : baseline	0.9989	0.9984	0.9985	0.99996
Over sample : tuned	0.9788	0.9696	0.9767	0.9975

49%	1%
1%	49%

PIPELINE


Hyperparameter tuning
Random Forest
MLP Classifier
Parameters tested
random_grid = {
'hidden_layer_sizes': [(50,), (50,50), (100,)],
'activation': ['logistic'],
'solver': ['sgd', 'adam'],
'alpha': [0.0001, 0.001, 0.01,0.1],
'learning_rate': ['constant','adaptive'],
'max_iter': [1000,1500,2000]
}
Best model
MLP_best = MLPClassifier(solver= 'adam', max_iter= 1500, learning_rate= 'adaptive', hidden_layer_sizes= (100,), alpha= 0.1, activation= 'logistic')

RESULTS

Hyperparameter tuning

Random Forest

MLP - Standard	Accuracy	Precision	Sensitivity	AUC
Under sample : baseline	0.8518	0.7963	0.8664	0.9258
Under sample : tuned	0.8696	0.8162	0.8898	0.9427

42%	8%
6%	44%

MLP Classifier

MLP - Standard	Accuracy	Precision	Sensitivity	AUC
Normal Sample	0.9978	0.5962	0.7067	0.4388
Normal Sample : tuned	0.9948	0.0269	0.0354	0.90835

99,46%	0.01%
0.51%	0.02%

MLP - Standard	Accuracy	Precision	Sensitivity	AUC
Over Sample	0.9974	0.9956	0.9984	0.4906
Over Sample : tuned	0.9787	0.9695	0.9767	0.9974

49%	1%
1%	49%

RESULTS

Hyperparameter tuning

Random Forest

44%	6%
5%	45%

Under sample (50%)

99,44%	0.03%
0.15%	0.37%

Normal sample (0.52%)

49%	1%
1%	48,888%

Over sample (50%)

MLP Classifier

42%	8%
6%	44%

99,46%	0.01%
0.51%	0.02%

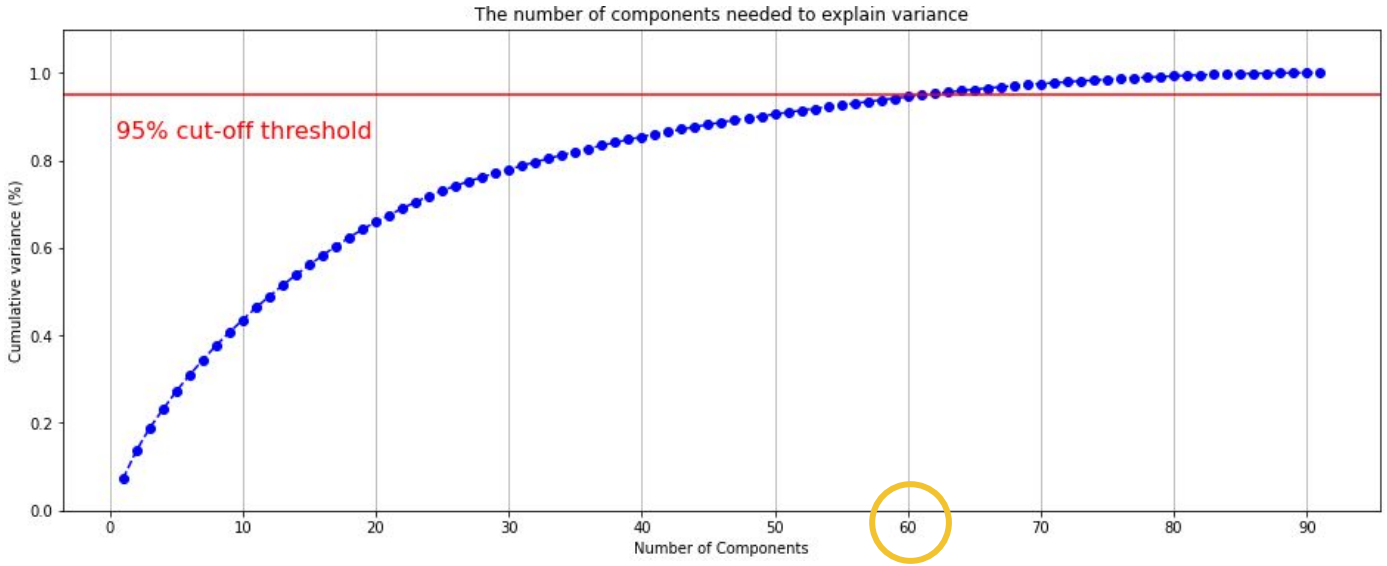
49%	1%
1%	48,891%

RESULTS

Random Forest

PCA

Sensitivity	Baseline	Tuned	+ PCA
Under	88,16%	89,02%	84,70%



APPROACHES TRIED

Classification

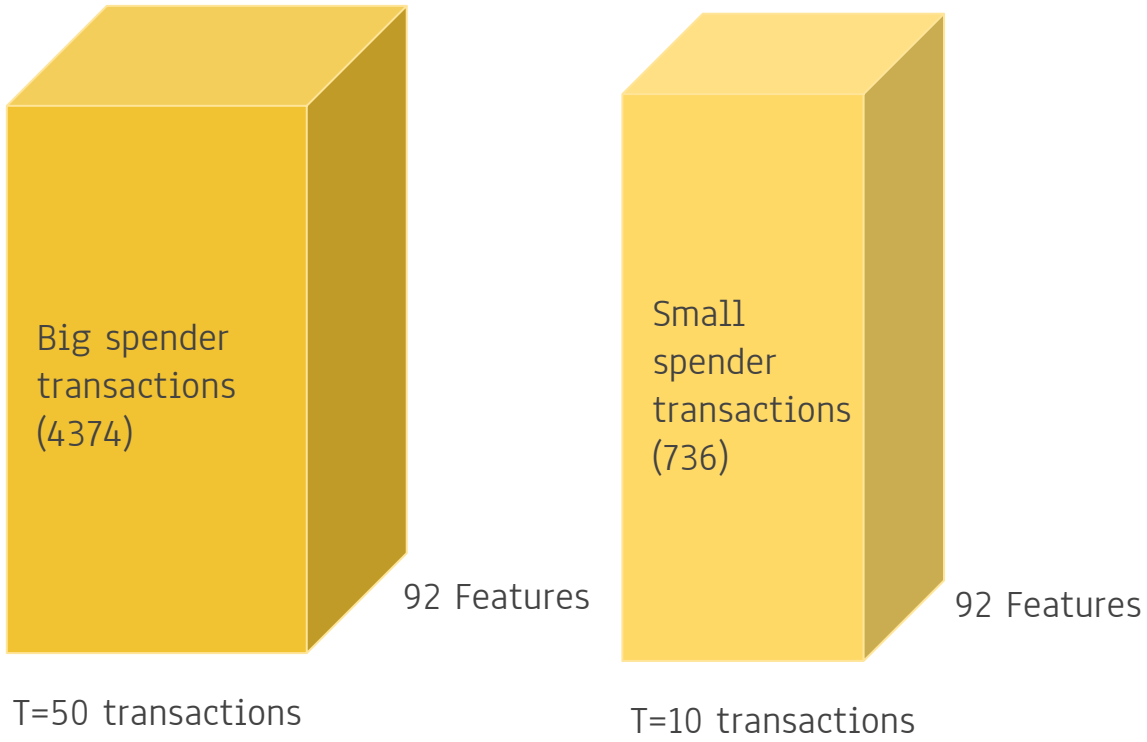
LSTM

LSTM FORMAT

Studied 2 “types” of cardholders:

- **Cardholder 1** (“Big Spender”) :
  Had a total amount of transactions of $411K over the 2 year period, with 4374 transactions

- **Cardholder 2** (“Small Spender”) :
  Had a total amount of transactions of $38K over the 2 year period, 736 transactions.



RESULTS

LSTM model implementation used default hyperparameter values.

For **Cardholder 1** ("Big Spender"), used timestep value T=50.

At default threshold=0.50, model performance is:

```
----------------------------------------------------
Accuracy: 0.998 , Sensitivity: 0.0, Precision: 0.0
----------------------------------------------------
```

When threshold=0.305, Sensitivity performance improves:

```
Accuracy: 0.669 , Sensitivity: 0.5, Precision: 0.0
```

For **Cardholder 2** ("Small Spender"), used timestep value T= 10

At default threshold=0.50, model performance is optimal for Accuracy:
Other threshold values yield suboptimal results.

```
----------------------------------------------
Accuracy: 1.00 , Sensitivity: 0., Precision: 0.
```

FINAL THOUGHTS

KEY TAKEAWAYS

CONCLUSION

Classification

1. The models that worked well on the under sample, work even better on the oversample.
2. The best model, on all type of samples is the **Random Forest**.
3. In this particular context of fraud, the random **OVER sample** worked best at identifying all the fraud cases (99,85% Sensitivity).
4. **PCA** does not mean the accuracy will increase.

LSTM

1. Although performance results were disappointingly low for Sensitivity, the LSTM gave **100% accuracy** for default threshold values and for varied timesteps.
2. Further analysis is needed to implement LSTM models on fraud data with extreme **imbalance** in the Fraud class.

IMPROVEMENTS

- Try PCA earlier in the process;
- Take the time to create bigger categories for the categorical variables;
- Test out other models;
- Try other under/over sampling methods.

LIMITS

- Time
- RAM Capacity :
  - Oversample : Couldn't converge the SMOTE-ENN method and couldn't use the full dataset
  - Gridsearch limits

Team 6

Model Performance of Different Machine Learning Models for Binary Sentiment Classification

Task

- Comparison of different model performances for binary sentiment classification of texts on the same dataset

- Use similar to identical preprocessing for each model except for the model in deep learning: TF-IDF

Dataset

IMDB Dataset of 50K Movie Reviews

- 1st column: Movie Review Text
- 2nd column: Sentiment label (positive or negative)

Descriptive Statistics

	review	sentiment
count	50000	50000
unique	49582	2
top	Loved today's show!!! It was a variety and not...	positive
freq	5	25000

sentiment	
negative	25000
positive	25000

Dataset

- Duplicates Deleted
- After duplicates are removed we have approximately 50% positive sentiments and 50% negative sentiments

Descriptive Statistics

	review	sentiment
count	49582	49582
unique	49582	2
top	One of the other reviewers has mentioned that ...	positive
freq	1	24884

sentiment	
negative	24698
positive	24884

Dataset: Train-Test Split

- Split into 70-30 train test split with a sample size of X
- We will be using 2 stratified sets, one with 1000 observations and another with 4000 observations
- After our split, we have similar split ratios: 50-50

Descriptive Statistics

	review	sentiment
count	1000	1000
unique	1000	2
top	When his elderly mother Emily (Jeanne Bates) i...	positive
freq	1	500

sentiment	review
negative	500
positive	500

	review	sentiment
count	4000	4000
unique	4000	2
top	When his elderly mother Emily (Jeanne Bates) i...	positive
freq	1	2000

sentiment	review
negative	2000
positive	2000

Dataset: Preprocess

TF-IDF transformation:

- TF - Term Frequency: Dampening function to reduce importance of a excessively frequent word occurrences
- IDF - Inverse Document Frequency: Decreasing function to down weigh commonly occurring words because they are statistically less discriminative (avoid spam)

Stop words:

- Removal of commonly occurring words completely

Lowercasing the words

AGGARWAL, C. H. A. R. U. C. (2018). Section 3.1. In *Recommender systems: The textbook* (pp. 145–146). essay, SPRINGER.

Evaluation Setup

- Set up a baseline: Naive bayes - simple, computationally light
- Develop multiple models to compare to the baseline:
  - Logistic Regression
  - KNN
  - Trees
  - Random Forest
  - Neural Networks
  - Berts
- Parameter tuning of the comparing models
- Base performance on accuracy since it is a classification problem

Model: Logistic Regression

Function: LogisticRegression from Sklearn

- Classification algorithm part of the GLM family : Link=logit [logit(z)=ln(z/1-z)]
- Takes transformed vectors consisting of words and their frequencies as the predictor variables and outputs a probability between 0 and 1
- We can then set a threshold to classify between the given positive or negative sentiment
- Accuracy : TP + TN / Total observations
- CV validation of “C” parameter which is the inverse of the regularization (smaller value means stronger regularization)

Model: Naive Bayes

Function: MultinomialNB from Sklearn

- Naive Bayes is used as the baseline
- We try to establish the probability of a sentiment given important words in the review
- This works particularly well when data is scarce and important keywords are easily identifiable

$$P(\text{sentiment}|\text{word}) = P(\text{word}|\text{sentiment}) * P(\text{sentiment}) / P(\text{word})$$

Example: amazing

Model: KNN

Function: KNeighborsClassifier from Sklearn

- We measure the distance between features of similar reviews
- When the sentences are converted into tokens, the feature transforms from a single sentence into multiples features that represent words
- The most similar sentences are then associated by the distance function and we can then compare the labels (the sentiment in our case)
- Euclidean distance is used as the default distance function in Sklearn
- Implemented as a majority vote

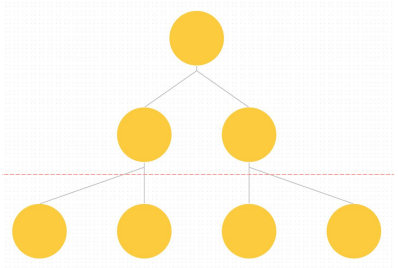
Example: Thriller movie

Model: Decision Tree Classifier

Function: DecisionTreeClassifier from Sklearn

This model partitions the data by creating splits. The decision to split is based on the impurity. It's to obtain homogenous split. There are different methods to calculate this impurity. The most popular one is the Gini impurity, lower is better.

For our model, we use the "Gini" impurity. The hyperparameter tuned is max_depth for the maximum depth of the tree using Grid Search. It will search over the values 4, 5 and 10.



Model: Random Forest Classifier

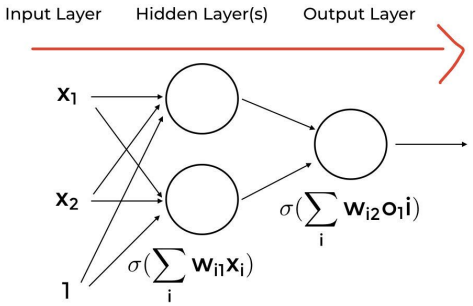
Function: RandomForestClassifier from Sklearn

It's an ensemble of decision trees using the bootstrap method. The bootstrap method let us sample an observation more than once. From the sample dataset, we create a tree by randomly using a subset of the covariates and do the split from them. We repeat the process multiple times. With all the trees built, we do the prediction based on the one with the highest mean probability.

For our model, we use the "Gini" impurity. The hyperparameter tuned is max_depth for the maximum depth of the tree using Grid Search. It will search over the values 4, 5 and 10.

Neural Network

Function: MLPclassifier from Sklearn



- Input layer = 10 000 + 1 (Max_feature = 10 000)
  - Takes in the TF-IDF features in the 10 001 input layers
- 3 Neural nets implemented: 1 hidden layer, 2 hidden layers and 3 hidden layers
  - Hidden layers weighted sums of their inputs followed by an activation fonction
  - If the sum of that node is past a certain threshold, it will pass it on to the next layer
- Output layer = 1 = the classification positive or negative
- CV was done on number of nodes in a layer, L2 penalty and learning rate
- Feed forward so left to right

Dataset for transformer

- Original dataset is 50 000 rows
- Duplicates are deleted
- We will be using 2 sets, one with 1000 observations and another with 4000 observations
- Split into 56% train, 20% for validation and 24% for test with stratified sampling on the sentiment

	1000	**4000**
train	560	2240
validation	200	800
test	240	960

Novelty: BERT

For this project, we have decided to tackle a new type of model that hasn't been seen in class: **BERT** models

The acronym stands for Bidirectional Encoder Representations from Transformers

Why BERT?

- It has a lot of documentation research paper, online projects
- It is a fairly recent approach to NLP: it was thought out in 2017 and created in 2018
- Its applications are almost exclusively related to NLP, which is a perfect match for the project we have constructed in sentiment analysis

Model: Transformers

Identifies the context in which a word gets its meaning in a sentence through a process called attention

Contrary to LSTM, the previous token isn't the only element that matters and the model can go back further back and forward in the sentence to find context in other tokens

This makes transformers particularly interesting in the field of translation: instead of translating word-by-word according to their position, the model translates given the context of the position in the sentence

It can be applied to sentiment analysis, for example with the word "great"

Model: BERT

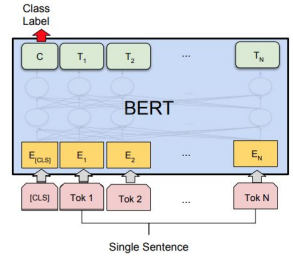
BERT is a transfer learning model; there are many pre-trained models available, but it is possible to finetune them on your desired dataset

The reason why models are pre-trained is that it takes an astronomical amount of time to train the model (such as with the Wikipedia Corpus), but it is then easier to finetune over a smaller dataset (ex: our 50 000 movie reviews)

The BERT model masks tokens in a sentence (about 15% of tokens in general) and does next sentence prediction

The process mentioned above helps BERT learn the context (it is a transformer-based model after all) for every word in a sentence

Literature Review



(b) Single Sentence Classification Tasks: SST-2, CoLA

Pre-training of Deep Bidirectional Transformers for Language Understanding

BERT has been proposed to improve the approach of fine-tuning pre-trained languages representations

Fine-tuning the model only requires an additional output layer, which makes it fast and usable in a variety of cases (few new parameters are learned)

By masking some tokens in the sentence, the models uses the words left-to-right and right-to-left to predict the masked word

We will be using BERT (BASE) and uncased, defined as having 12 layers and words are in lowercase, for our project

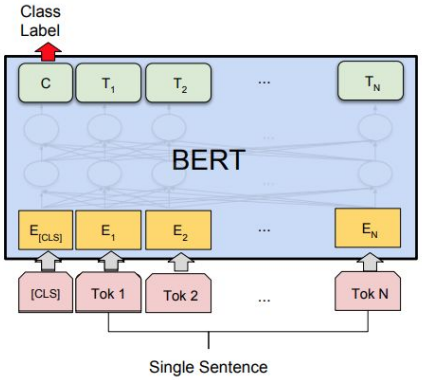
Literature Review

Fine-tuning is straightforward since the self-attention mechanism in the Transformer allows BERT to model many downstream tasks—whether they involve single text or text pairs—by swapping out the appropriate inputs and outputs.

For each task, we simply plug in the task-specific inputs and outputs into BERT and finetune all the parameters end-to-end:

Input: a degenerate text-∅ pair in text classification

Output: [CLS] representation is fed into an output layer for classification, such as sentiment analysis.



(b) Single Sentence Classification Tasks: SST-2, CoLA

Approach

- We used scikit-learn for these models: Naive bayes, Logistic regression, Decision tree, Random forest, Knn, Neural Network. Hugging Face for the Bert model.
- Scikit has the "pipeline" module. Hugging Face has the "Trainer" class.
- In general, we prioritize a pipeline framework by listing all the steps needed and execute them together.

```
# scikit
pipe_model_nb = make_pipeline(text.TfidfVectorizer(lowercase=True, stop_words="english"),
                              naive_bayes.MultinomialNB())
# huggingface
trainer = Trainer(
    model,
    training_args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["test"],
    tokenizer=tokenizer,
    compute_metrics=compute_metrics,
```

Results

For the first 6 models using CV, we splitted the data as 70% for the train and 30% on the test.
For the transformer, we used 56% train, 24% for validation and 20% for test.

models/ observations	Naive bayes	Logistic regression	Decision tree	Rando m forest	Knn	Neural Network 1 layer	Neural Network 2 layer	Neural Network 3 layer	Pretrained Bert uncase
1000	84.333%	83.000%	70.667%	82.667%	70.000%	83.000%	83.333%	83.333%	**89.167%**
4000	84.917%	84.750%	69.417%	82.333%	74.417%	82.333%	82.833%	83.167%	**91.563%**

Best Hyperparameters

models/ observations	Naive bayes	Logistic regression	Decision tree	Random forest	Knn	Neural Network 1 layer	Neural Network 2 layer	Neural Network 3 layer	Pretrained Bert uncase
1000	NA	C=100	max_depth= 5	max_depth= 4	N_neighbors = 15	Hidden layer size: 1000 Alpha: 0.1 Learning_rate : 0.001	Hidden layer size: (100,100) Alpha: 0.01 Learning_rate : 0.001	Hidden layer size: (25,100,100) Alpha: 0.01 Learning_rate : 0.01	NA
4000	NA	C=1	max_depth= 10	max_depth= 10	N_neighbors = 15	Hidden layer size: 100 Alpha: 0.1 Learning_rate : 0.001	Hidden layer size: (25,25) Alpha: 0.001 Learning_rate : 0.01	Hidden layer size: (100,100,50) Alpha: 0.01 Learning_rate : 0.001	NA

Future direction

How one can extend your work?

- Beside Bert, we can use other pretrained models. From this website, XLNet provides better performance and it's available on Hugging Face. https://paperswithcode.com/sota/sentiment-analysis-on-imdb?p=distilbert-a-distilled-version-of-bert

How one can improve the performance of your model?

- We currently used a subset of the data due to computing limits. With better hardwares from cloud providers such as AWS, Google cloud and Azure, we can use the whole dataset.
- Currently the number of epochs is 3. This is a hyperparameter we can tune through Hugging Face with the Trainer class. https://huggingface.co/docs/transformers/main_classes/trainer

Appendix - References

Grid search:
https://scikit-learn.org/stable/modules/grid_search.html
Decision tree:
https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier
Random forest:
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier

Neural Network:

https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414#:~:text=Modeled%20loosely%20on%20the%20human,them%20in%20only%20one%20direction

https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html?highlight=mlpclassifier#sklearn.neural_network.MLPClassifier

Appendix - References

Naive Bayes:

https://iq.opengenus.org/text-classification-naive-bayes/

KNN:

https://www.actonscholars.org/wp-content/uploads/Sentiment-Analysis-With-KNN-Algorithm.pdf

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Transformers:

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

BERT:

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
https://huggingface.co/bert-base-uncased

Team 7

Face Mask Detection

LOIC MOREAU
EMILIE TRUCA

Task and Dataset

TASK

- Supervised learning
- Detect masks on images

DATASET

- Small : 853 images
- 3 classes : ***wear a mask***, ***wear incorrectly a mask*** and ***do not wear a mask***




Literature review

One-stage (SSD)

VGG-16 (2014)

22 FPS

- CNN trained on ImageNet dataset (14 million images and 22,000 categories)

=> **One of the top-performing models submitted for the ILSVRC**

Two-stage (Faster R-CNN)

R-CNN (2014)

40-50s / img

- 3 models
- Apply a CNN 2,000 times per image for 2,000 Regions of Interest

=> **Expensive and slow**

Fast R-CNN (2015)

2s / img

- Single model
- Apply a CNN once per image for 2,000 RoI

=> **Faster than R-CNN**

Faster R-CNN (2015)

0.2s / img

- Use “Region Proposal Network” (RPN)

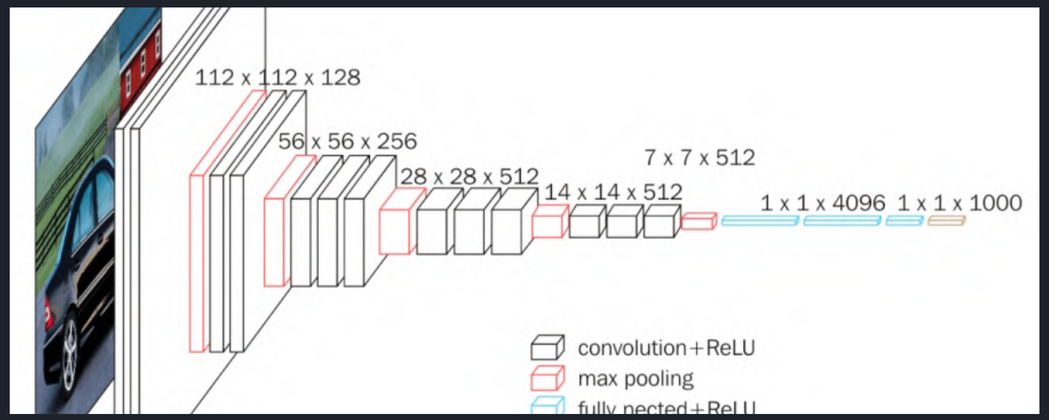
=> **Faster than Fast R-CNN**

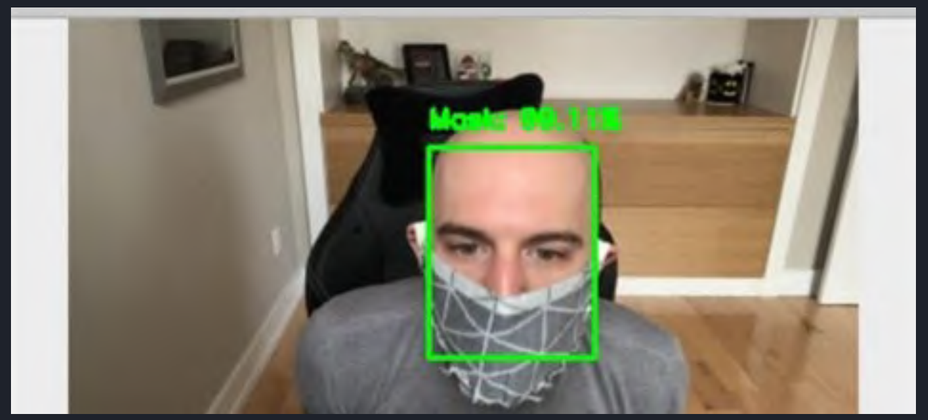
1. Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.
2. Girshick, Ross. "Fast r-cnn." Proceedings of the IEEE international conference on computer vision. 2015.
3. Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems. 2015.
4. W. Liu, D. Anguelov, D. Erhan, C. Szegedy and S. Reed, "SSD: Single shot multibox detector", 2015.

Approach

Model 1 - SSD + Classification

- Keras Sequential model with SSD architecture
- SSD model is considered a simple model
- Fast training time and easy to code
- SSD treats object detection as a simple regression problem



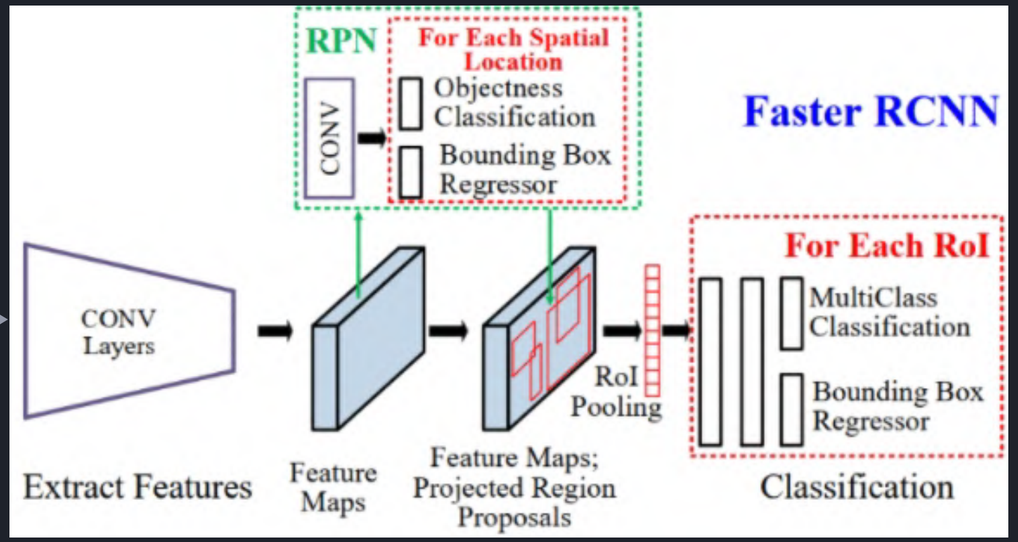


Approach

Model 2 - Faster R-CNN

```
model = torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)
```

- Pre-trained on COCO (Common Objects in Context) dataset
- Replace the final classification layer with 3 classes



Evaluation Setup

- Mean Average precision (mAP) : combine bounding box detection and classification accuracy
- Data split : 753 train / 50 val / 50 test
- Try 3 CNN : MobileNet-small, MobileNet-large, ResNet-50
- Train for 10 epochs, keep the model with best validation mAP

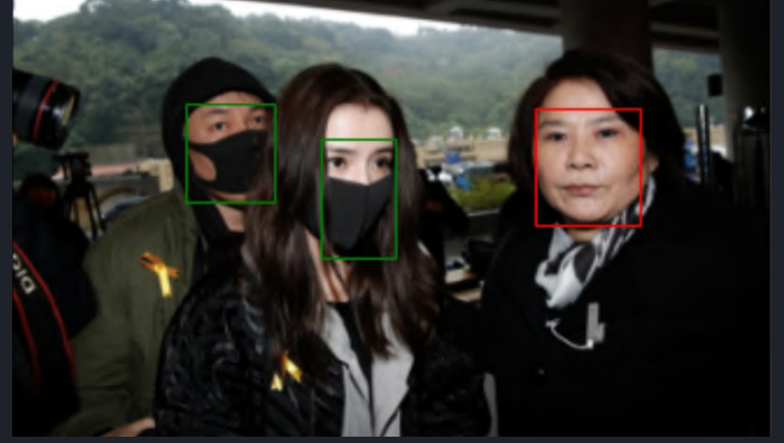
Results

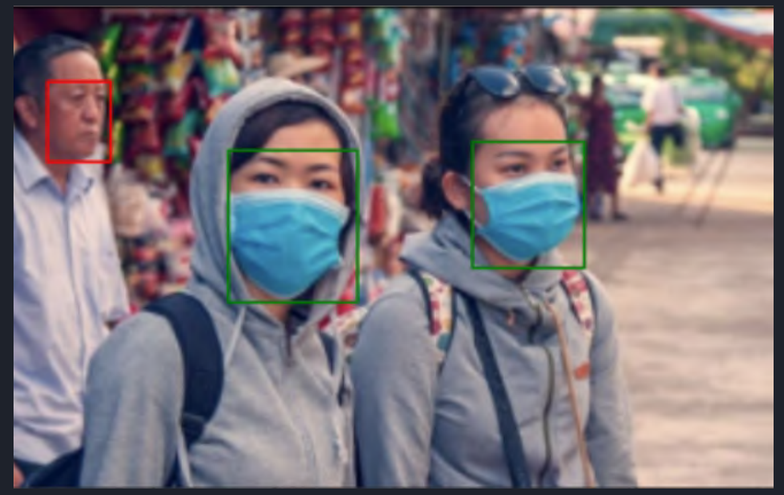
Baseline Model

CNN	Binary Cross Entropy	Classification Accuracy
VGG-16	0.25	0.61

Faster R-CNN

CNN	Mean Average Precision	Average Precision (IoU=0.5)
MobileNetV3-Large 320 FPN	0.466	0.779
MobileNetV3-Large FPN	0.476	0.758
Resnet-50 FPN	0.655	0.947





Future Direction

- More Hyper-parameter tuning, with more time

- Our dataset is small and not diverse (mostly asian people)

  => try our approach on a larger, more diverse dataset

- Try our approach on video data

  => real-time model

Team 8

Flight Delay and Prediction Based on Machine Learning Methods

Gita Gonoody

Master of business intelligence student

April/06/ 2022

Objective: Multi-class classification predictive modeling

Predicting flight delay, on time and earliness arrival

- Trying different supervised Multi-class ML methods to find the model with highest performance in order to predict each class. Based on our case study and data set.

- Data source: kaggle

Importance and problem :

❑ Importance:

- Passengers
- Airlines
- Airports
- Environment

❑ Problem:

- Delay propagation(How: airlines , departure city)
- Root delay and Cancellation(when and where , reasons and sources)

❑ Different aspects

- Temporal (month, day of week, time of day(hours)
- Weather (Temperature , windspeed , visibility,…)
- Spatial (city, airport,…)
- Planning (flight and airline)

source: A Review on Flight Delay Prediction, Alice Sternberg, Jorge Soares,.. April 2021

HEC MONTRÉAL

Problem and Dataset description

✓ From Kaggle :The information for flights delay between the all airports in unites States in 2015.

- Features: actual arriving and schedule arriving time, airlines, departure city . Departure delay , distances ,month ,week of the day
- Target variable: Y :Delay

✓ Climate information are added from NOAA .

- Features: temperature ,visibility, wind speed,… for departure and arrival city

HEC MONTRÉAL

Statistics related to dataset:

- ✓ More than 5million observations in the original data set
- ✓ Atlanta international airport as destination , most important hub in united state, more than 300000 observations
- ✓ 15 most frequents departure cities that have high number of flight to this airport (more than 5000 flights per year)

Chicago	New York	Orlando	Houston	Ft. Lauderdal	Tampa	Dallas-For	Philadelphia
10758	9877	8162	7945	7384	7040	6980	6540
Arlington	Charlotte	Jacksonvil	Baltimore	Los Angeles	New Orleans	Miami	
6357	6273	6202	5769	5646	5567	5125	

- ✓ 105000 observation from 15 cities in 2015
- ✓ All months and days and hours except October
- ✓ Count in each class Delay:
  - ❑ 0 : 0.530371 (-14< ARRIVAL_DELAY < 10) , median -6
  - ❑ -1 : 0.264325 (-59< ARRIVAL_DELAY < -15) , median -20
  - ❑ 1 0.205304 (11 < ARRIVAL_DELAY < 1184) , median 32
- ✓ data split:
  - ❑ Train 60%
  - ❑ Validation 20%
  - ❑ Test 20%

Data management:

- Data cleaning :
  - ✓ missing value(Baltimore)
  - ✓ extreme value(9999.9)
- Data transformation :
  - ✓ delay :convert continuous variable (Arrival delay) to categorical variable (Delay)(-1,0,1)
    - Arrival delay was calculated based on the difference between schedule and actual arrival time
  - ✓ Extract departure hour from departure time
  - ✓ Transform Airline (around 10) and city (around 15) categorical variables to dummies
  - ✓ Standardization of continuous variables

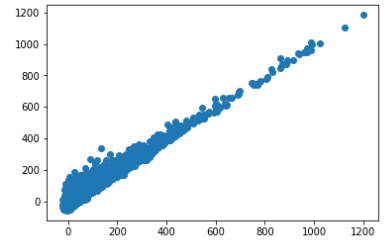
HEC MONTRÉAL

Statistics related to dataset:

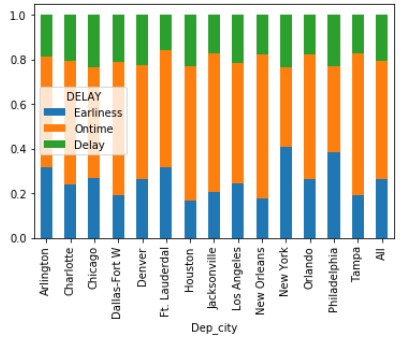
- ✓Departure delay and delay
- ✓Distance and delay
- ✓Hours and delay
- ✓City and delay
- ✓Airlines and delay
- ✓….

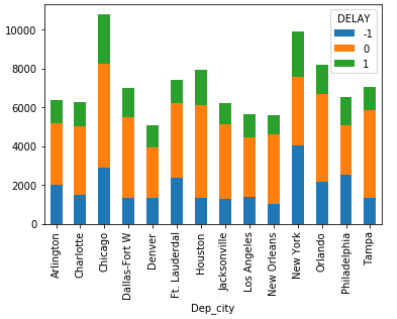
Statistics related to dataset:

- ✓ Positive correlation (0.94%)between departure delay and arrival delay

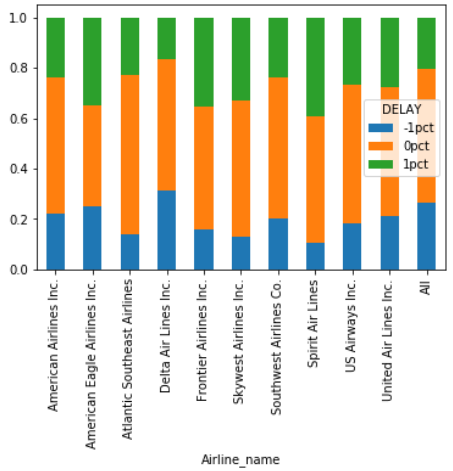


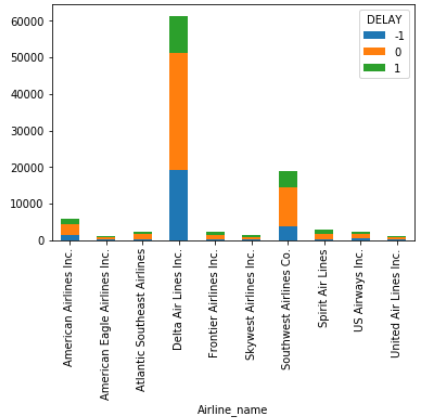
- ✓ Departure city and delay



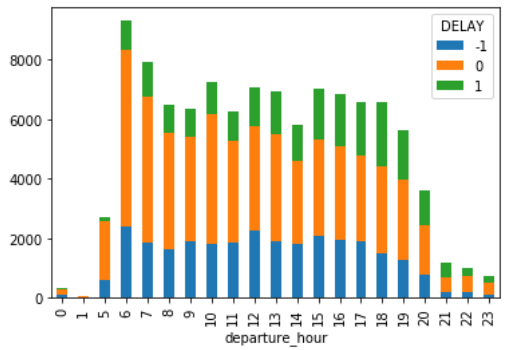


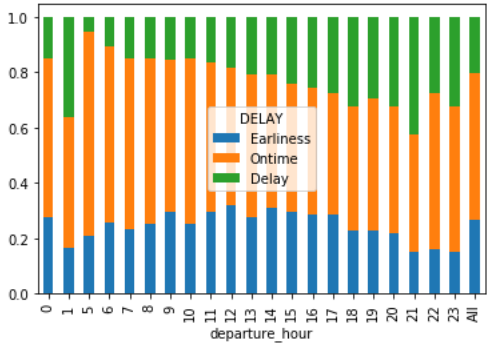
- ✓ Airline and delay





- ✓ Hours and delay





In [382]:

Literature review:

- ✓ A Review on Flight Delay Prediction, Alice Sternberg, Jorge Soares,.. April 2021
- ✓ Development of a predictive model for on-time arrival flight of airliner by discovering correlation between flight and weather data, Etani, Noriko, Journal of Big Data, 2019
  - Random Forest Classifier with weather data and Gradient Boosting Classifier without weather data are adopted as the predictive model
- ✓ Characterization and prediction of air traffic delays J. J. Rebollo and H. Balakrishnan.. , July 2014. ISSN 0968-090X
  - Random Forest

HEC MONTRÉAL

Approach:

- Machine learning methods:learn from train set and predict unseen dataset
- Supervised learning (Labels)
- Multi-class classification(3 categories)
- Apply different Machine learning methods for Multi-class classification :
  - ❑ Desicion tree classifier
  - ❑ RandomForestClassifier
  - ❑ GradientBoostingClassifier
  - ❑ KNeighborsClassifier
  - ❑ SVM
  - ❑ Neural Network
  - ❑ Deep learning

HEC MONTRÉAL

Evaluation Setup:

- Baseline model
  - ✓ RandomForestClassifier
  - ✓ GradientBoostingClassifier
  - ✓ NN
- Data split:
  - ➢ 60% Train, 20% Validation, 20% test
- hyper-parameter tuning
  - ❑ RandomForestClassifier
    - ➢ n_estimators, max_depth
  - ❑ GradientBoostingClassifier
    - ➢ n_estimators, learning_rate
  - ❑ NN
    - ✓ hidden layers and nodes

HEC MONTRÉAL

Results

list	Method	Accuracy on validation_set	tuning	Stability:accuracy on test_set
1	Majority class prediction	53.88%		
2	GaussianNB	55%		
3	MultinomialNB	54%		
4_1	Desicion tree classifier	63.26%	criterion='gini'	
4_2		63.04%	entropy	
5_1	RandomForestClassifier	72.99%	n_estimators=200, max_depth=40	
5_2		73.08%	n_estimators=300, max_depth=40	**73.10%**
5_3		72.96%	n_estimators=400, max_depth=40	
5_4		72.99%	n_estimators=400, max_depth=50	
5_5		73.02%	n_estimators=500, max_depth=50	
5_6		73.03%	n_estimators=700, max_depth=50	
6_1	GradientBoostingClassifier	70%	n_estimators=100 ,max_depth=3	
6_2		71.37%	n_estimators=200, learning_rate=0.1	
6_3		73%	n_estimators=200, learning_rate=0.5	
6_4		72.31%	n_estimators=200, learning_rate=1	
6_5		72.92%	n_estimators=300, learning_rate=0.5	
6_6		72.94%	n_estimators=400, learning_rate=0.5	73.46%
7_1	KNeighborsClassifier	56%	n_neighbors=5	
7_2		57.29%	n_neighbors=10	
8_1	SVM	63.64%	OVR	
9_1	Neural Network	71%	1 hidden layers of 100 nodes	71.94%
9_2	Deep learning	69%	3 hidden layers and 100 nodes	
9_3		68%	hidden layers and 100 nodes	

HEC MONTRÉAL

Future Direction

- ✓ More model LSTM
  - Based on Flight Delay Prediction Based on Aviation Big Data and Machine Learning, Guan Gui; Fan Liu; Jinlong Sun, Jan. 2020
  - accuracy (90.2%)
- ✓ Add more variables
  - Airplane model
  - Different departure cities
  - Airports
- ✓ Optimizing hyperparameters
- ✓ Over sampling or under sampling

HEC MONTRÉAL

- Questions?

Team 9