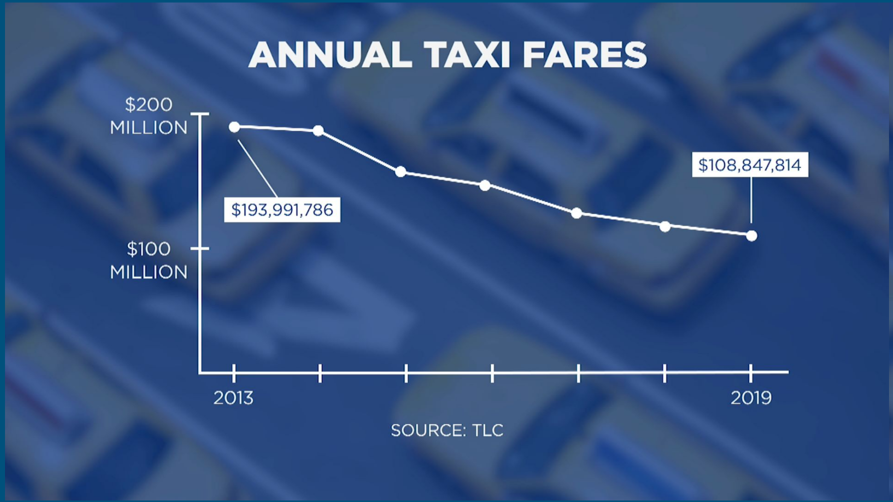
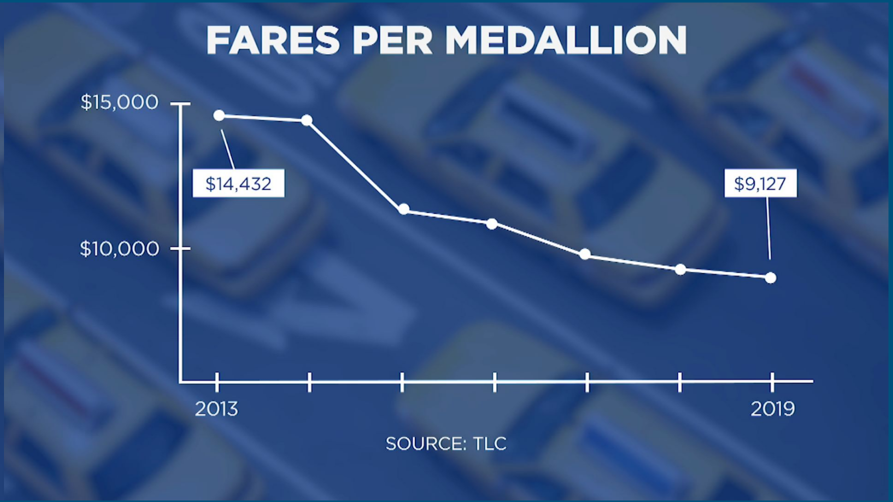
Mini Project 3

Munachiso Nwadike, Romeno Wenogk
and Takumi Miyawaki

Problem Statement



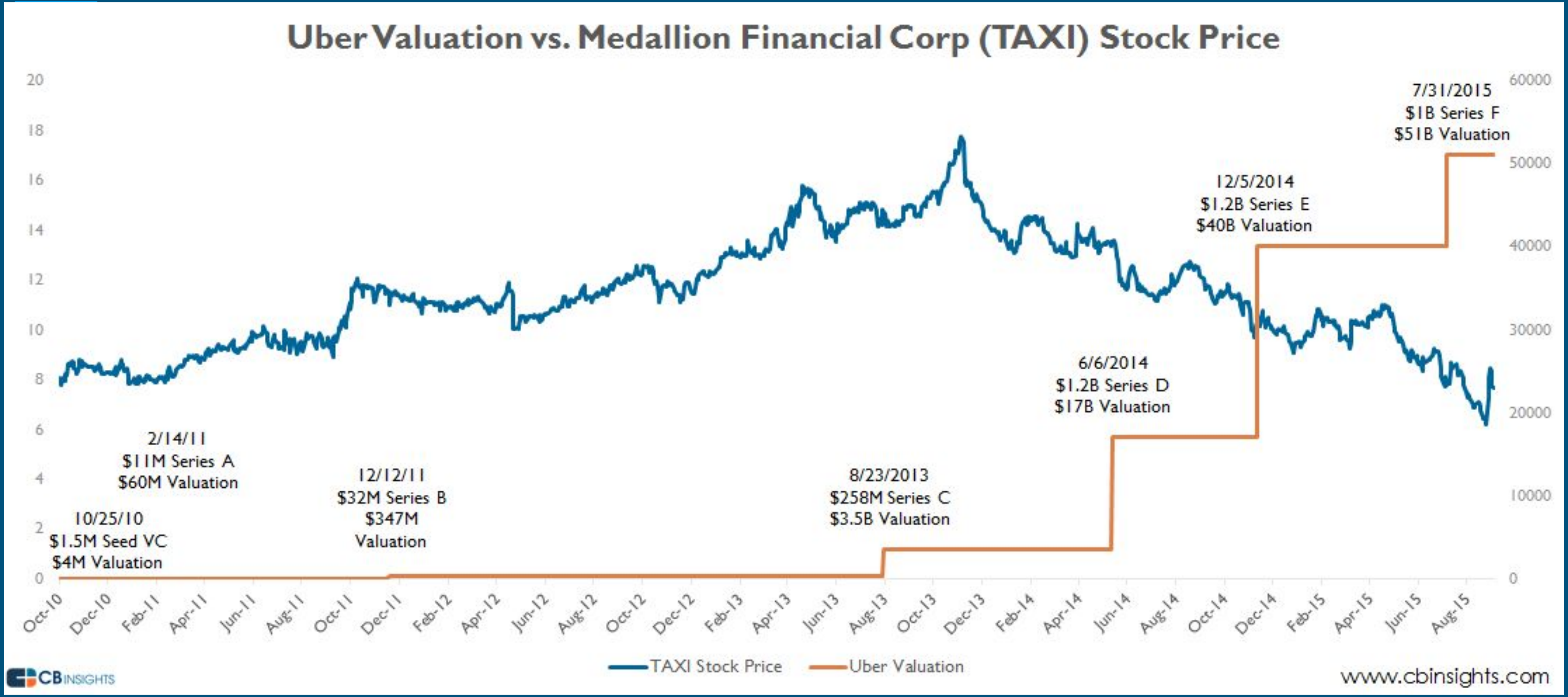


Solution

Standard Metered Fare

- **$2.50** initial charge.
- Plus **50 cents** per 1/5 mile when traveling above 12mph or per 60 seconds in slow traffic or when the vehicle is stopped.
- Plus **50 cents** MTA State Surcharge for all trips that end in New York City or Nassau, Suffolk, Westchester, Rockland, Dutchess, Orange or Putnam Counties.
- Plus **30 cents** Improvement Surcharge.
- Plus **50 cents** overnight surcharge 8pm to 6am.
- Plus **$1.00** rush hour surcharge from 4pm to 8pm on weekdays, excluding holidays.
- Plus New York State Congestion Surcharge of **$2.50** (Yellow Taxi) or **$2.75** (Green Taxi and FHV) or **75 cents** (any shared ride) for all trips that begin, end or pass through Manhattan south of 96th Street.

A first Look - Uber Vs. Taxis



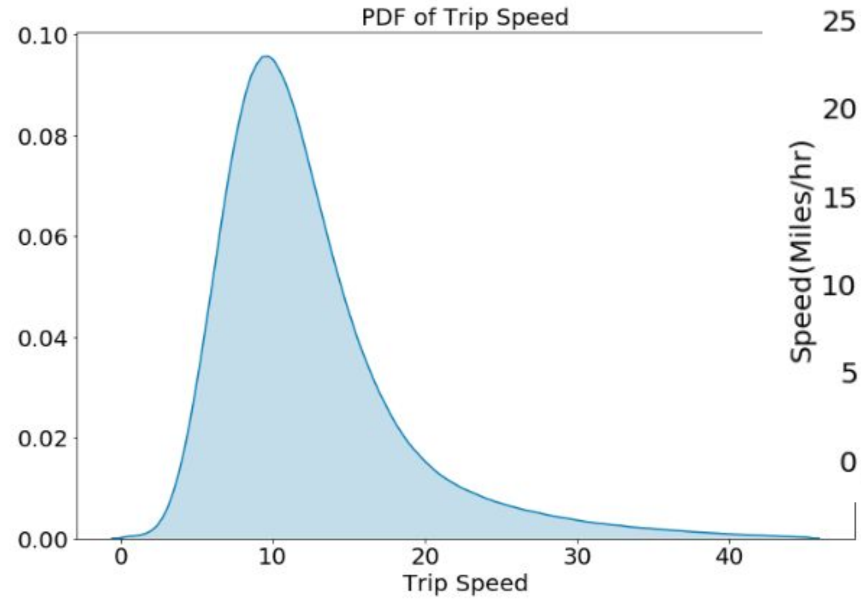
Raw Data

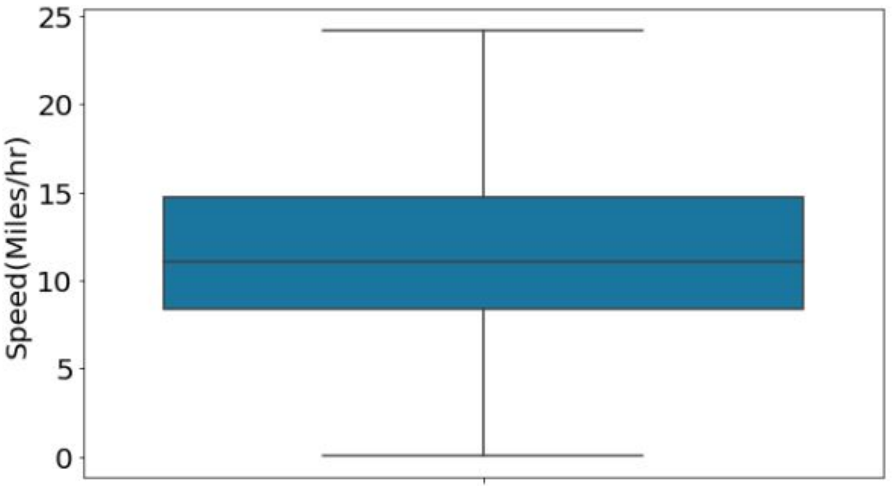
Outliers!

- Example 1: 99.9th percentile value of speed is 45.31miles/hr. 100th percentile value of a speed is 192 Million miles/hr.
- Example 2: ->

Data Cleaning (Speed)

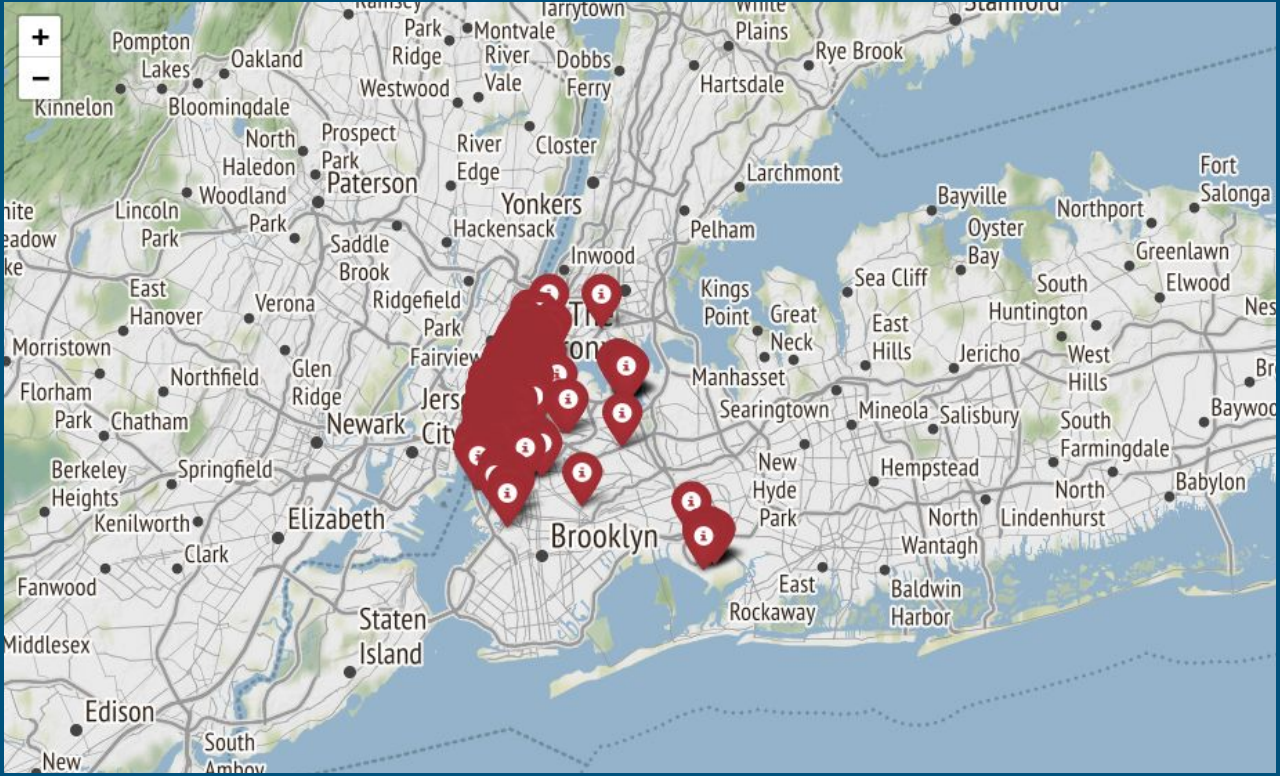
```
new_frame_cleaned = new_frame_cleaned[(new_frame_cleaned.speed>0) & (new_frame_cleaned.speed<45.31)]
print("Speed Outliers removed")
print("-"*35)
```





- 30 clusters, 4464 time bins

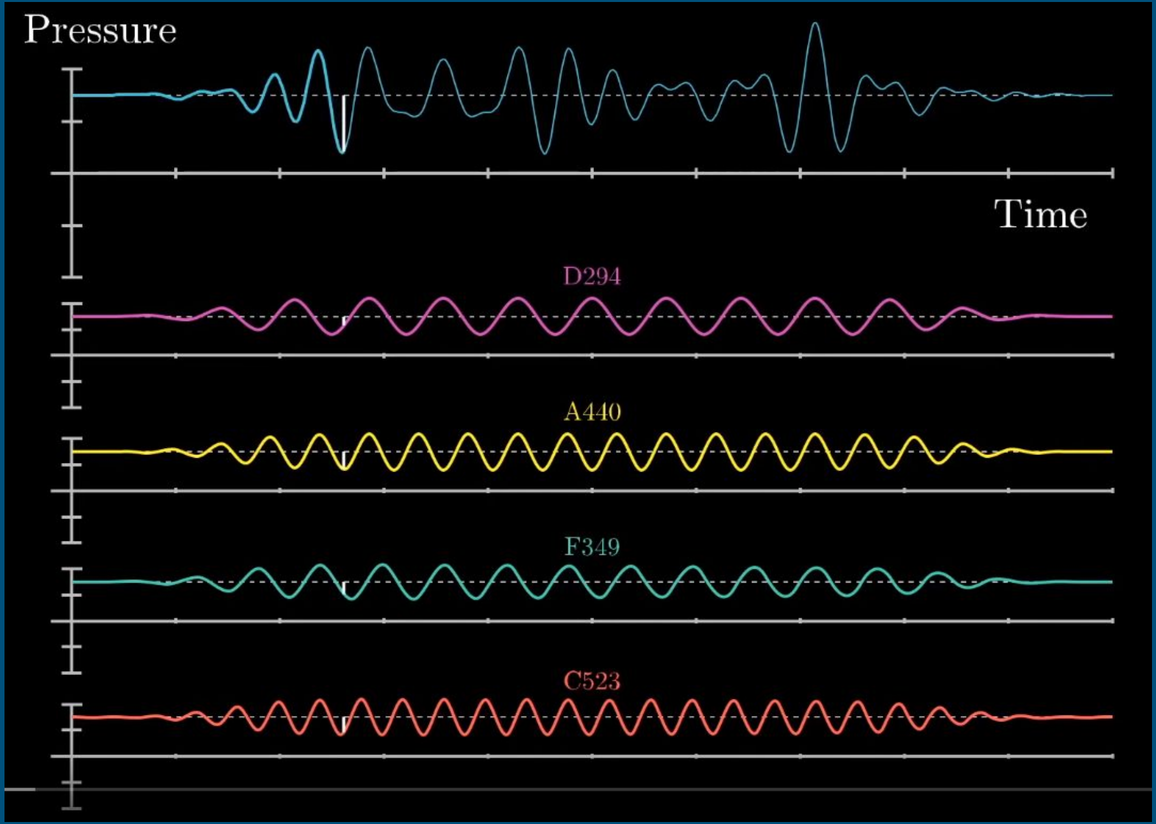
Data Cleaning (location)

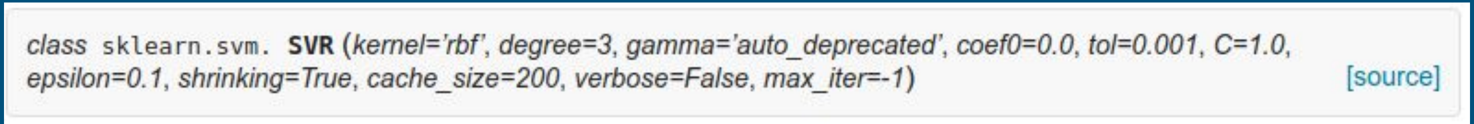


Data Features

[Goal: Predict Speed at Given Pickup Location]

- Previous Pickups
- Day of the week
- Location of the Pickup cluster (Lat, Long)
- *Fourier Frequency & Amplitude?*

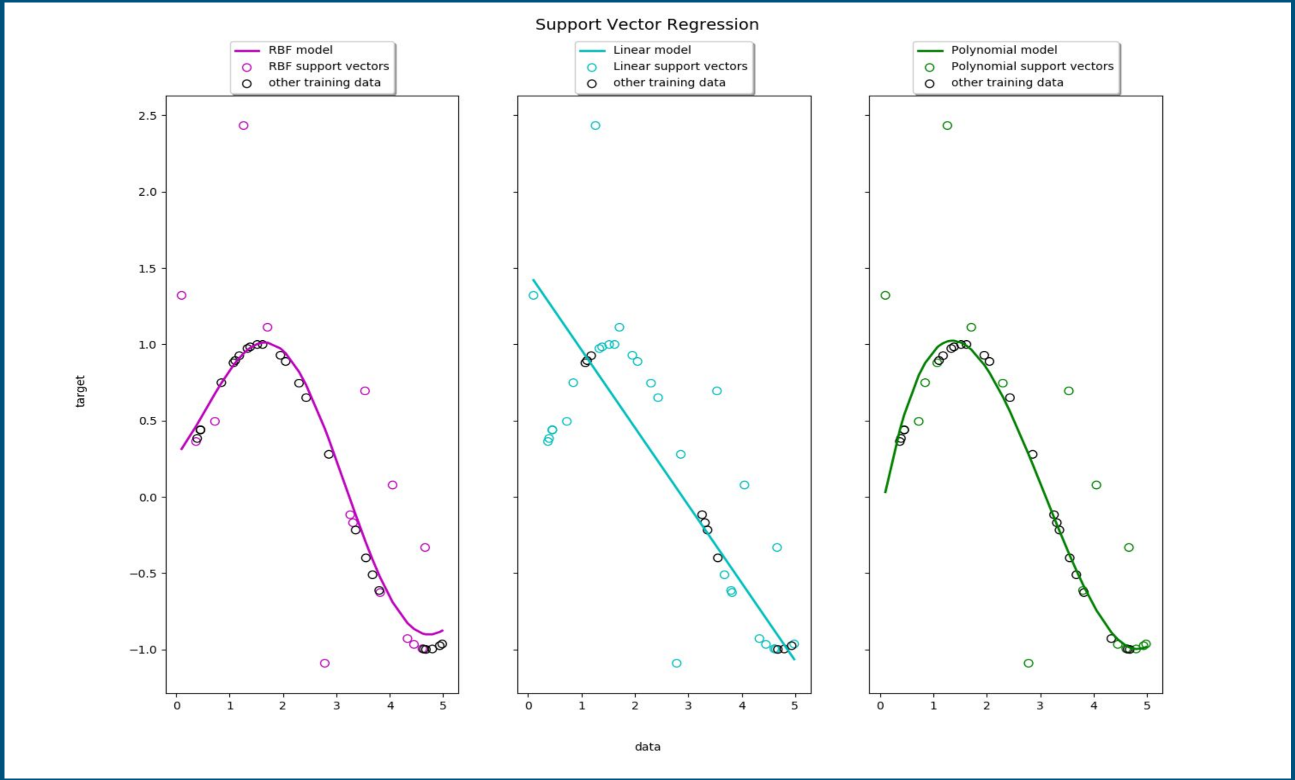




SV-Regression

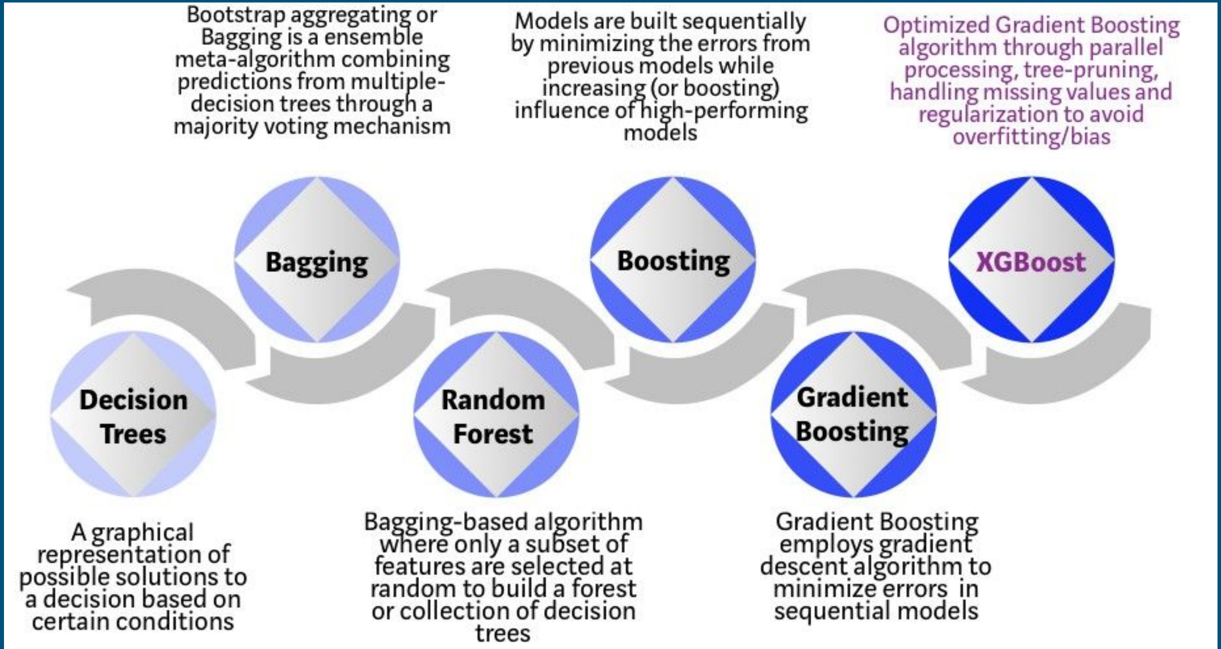
Similar to SVC.

Differs from other regression models as it is non-parametric [relies on Kernel functions]

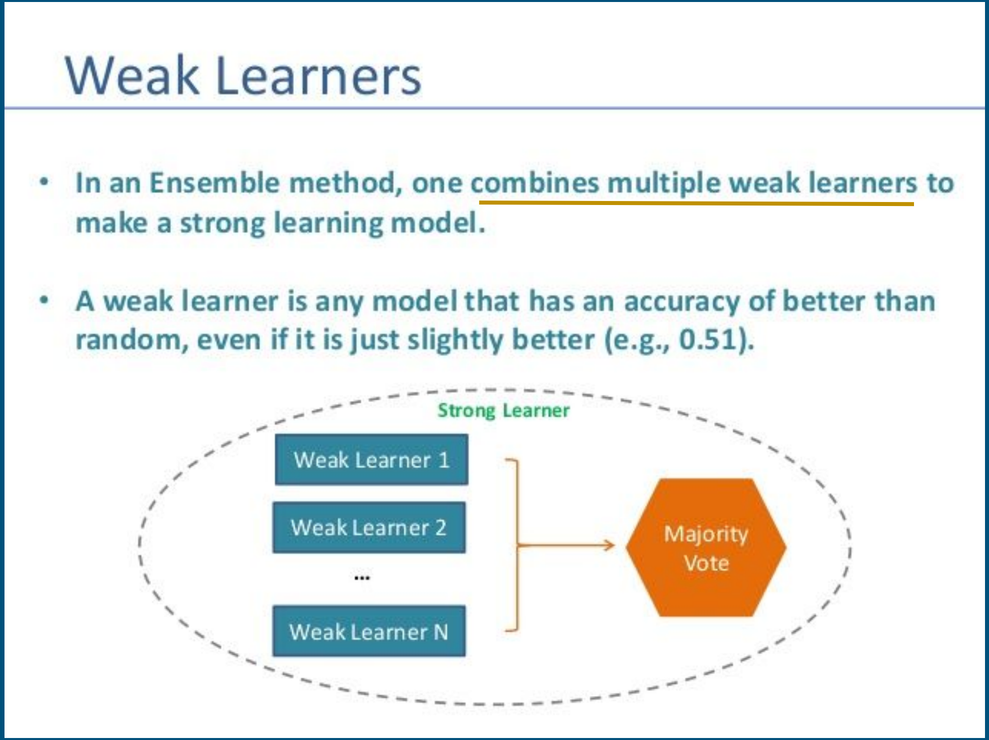


XGBoost

- Extreme Gradient Boosting
- Boosting -> new models added to correct the errors made by existing models. Models added sequentially.
- Gradient boosting -> gradient descent algorithm to minimize the loss when adding new models.



Ensemble Learning



XGBoost

XGBoost Regressor

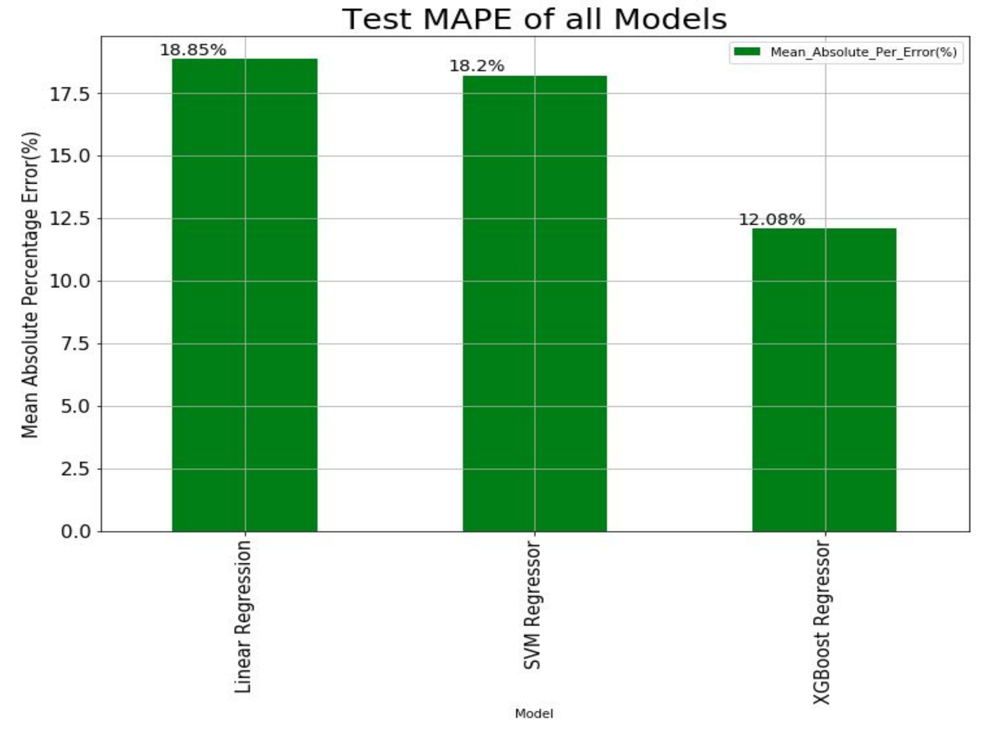
```
def xgboost_reg(train_data, train_true, test_data, test_true):
    #hyper-parameter tuning
    hyper_parameter = {"max_depth":[1, 2, 3, 4], "n_estimators":[40, 80, 150, 600]}
    clf = xgb.XGBRegressor()
    best_parameter = GridSearchCV(clf, hyper_parameter, scoring = "neg_mean_absolute_error", cv = 3)
    best_parameter.fit(train_data, train_true)
    estimators = best_parameter.best_params_["n_estimators"]
    depth = best_parameter.best_params_["max_depth"]

    #applying xgboost regressor with best hyper-parameter
    clf = xgb.XGBRegressor(max_depth = depth, n_estimators = estimators)
    clf.fit(train_data, train_true)
    train_pred = clf.predict(train_data)
    train_MAPE = mean_absolute_error(train_true, train_pred)/ (sum(train_true)/len(train_true))
    train_MSE = mean_squared_error(train_true, train_pred)
    test_pred = clf.predict(test_data)
    test_MAPE = mean_absolute_error(test_true, test_pred)/ (sum(test_true)/len(test_true))
    test_MSE = mean_squared_error(test_true, test_pred)

    return train_MAPE, train_MSE, test_MAPE, test_MSE
```

Results

	Model	TrainMAPE(%)	TrainMSE	TestMAPE(%)	TestMSE
0	Linear Regression	11.772115	9.662512	19.533328	10.811107
1	SVM Classifier	10.475732	8.884453	18.694722	10.301545
2	XGBoost Regressor	10.801919	8.556740	12.195017	7.416899



	Model	Mean_Absolute_Per_Error(%)
0	Linear Regression	19.5333
1	SVM Regressor	18.6947
2	XGBoost Regressor	12.195

Resources

- https://blog.goodaudience.com/taxi-demand-prediction-new-york-city-5e7b12305475
- https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d
- https://www.datacamp.com/community/tutorials/xgboost-in-python