Loading all train and test data files

In [1]:

```
import pandas as pd
Train_DF = pd.read_csv('Train_DF.csv')
Test_DF = pd.read_csv('Test_DF.csv')
```

In [2]:

```
with open("train_TruePickups_flat.txt","r") as file:
    train_TruePickups_flat=file.read().split()
```

In [3]:

```
with open("test_TruePickups_flat.txt","r") as file:
    test_TruePickups_flat=file.read().split()
```

In [4]:

```
train_TruePickups_flat = list(map(int, train_TruePickups_flat))
test_TruePickups_flat = list(map(int, test_TruePickups_flat))
```

Using Linear Regression

In [5]:

```
#standardizing the data
from sklearn.preprocessing import StandardScaler
train_std = StandardScaler().fit_transform(Train_DF)
test_std = StandardScaler().fit_transform(Test_DF)
```

In [6]:

```
from sklearn.linear_model import SGDRegressor
from sklearn.model_selection import GridSearchCV
#hyperparameter tuning
clf = SGDRegressor(loss = "squared_loss", penalty = "l2")
values = [10**-4,10**-3,10**-2,10**-1,1,10,100,1000]
hyper_parameter = {"alpha": values}
best_parameter = GridSearchCV(clf, hyper_parameter, scoring = "neg_mean_absolute_error", cv = 3)
best_parameter.fit(train_std, train_TruePickups_flat)
alpha = best_parameter.best_params_["alpha"]
```

In [7]:

```
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
clf = SGDRegressor(loss = "squared_loss", penalty = "l2", alpha = alpha)
clf.fit(train_std, train_TruePickups_flat)
train_pred = clf.predict(train_std)
lr_train_MAPE = mean_absolute_error(train_TruePickups_flat, train_pred)/
(sum(train_TruePickups_flat)/len(train_TruePickups_flat))
lr_train_MSE = mean_squared_error(train_TruePickups_flat, train_pred)
test_pred = clf.predict(test_std)
lr_test_MAPE = mean_absolute_error(test_TruePickups_flat, test_pred)/ (sum(test_TruePickups_flat)/l
en(test_TruePickups_flat))
lr_test_MSE = mean_squared_error(test_TruePickups_flat, test_pred)
```

In [8]:

```
print(" Using Logistic regression ")
print("train_MAPE:",lr_train_MAPE)
print("train_MSE:",lr_train_MSE)
```

```
print("test_MAPE:", lr_test_MAPE)
print("test_MSE:", lr_test_MSE)
```

```
 Using Logistic regression
train_MAPE: 0.13232113604281714
train_MSE: 295.7725799623722
test_MAPE: 0.13126781320135156
test_MSE: 265.5542883192399
```

Using Random Forest Regressor

In [9]:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import RandomizedSearchCV
values = [10, 40, 80, 150, 600, 800]
clf = RandomForestRegressor(n_jobs = -1)
hyper_parameter = {"n_estimators": values}
best_parameter = RandomizedSearchCV(clf, hyper_parameter, scoring = "neg_mean_absolute_error", cv =
3)
best_parameter.fit(Train_DF, train_TruePickups_flat)
estimators = best_parameter.best_params_["n_estimators"]
```

```
/home/ubuntu/.local/lib/python3.6/site-packages/sklearn/model_selection/_search.py:266:
UserWarning: The total space of parameters 6 is smaller than n_iter=10. Running 6 iterations. For
exhaustive searches, use GridSearchCV.
  % (grid_size, self.n_iter, grid_size), UserWarning)
```

In [10]:

```
rf = RandomForestRegressor(n_estimators = estimators, n_jobs = -1)
rf.fit(Train_DF, train_TruePickups_flat)
```

Out[10]:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=800, n_jobs=-1,
                      oob_score=False, random_state=None, verbose=0,
                      warm_start=False)
```

In [11]:

```
train_pred = rf.predict(Train_DF)
rf_train_MAPE = mean_absolute_error(train_TruePickups_flat, train_pred)/
(sum(train_TruePickups_flat)/len(train_TruePickups_flat))
rf_train_MSE = mean_squared_error(train_TruePickups_flat, train_pred)
```

In [12]:

```
test_pred = rf.predict(Test_DF)
rf_test_MAPE = mean_absolute_error(test_TruePickups_flat, test_pred)/ (sum(test_TruePickups_flat)/l
en(test_TruePickups_flat))
rf_test_MSE = mean_squared_error(test_TruePickups_flat, test_pred)
```

In [13]:

```
print(" Using Random Forest:  ")
print("train_MAPE:",rf_train_MAPE)
print("train_MSE:",rf_train_MSE)
print("test_MAPE:", rf_test_MAPE)
print("test_MSE:", rf_test_MSE)
```

```
 Using Random Forest:
train_MAPE: 0.0490885021833117
train_MSE: 49.07482649474671
```

```
test_MAPE: 0.11756638382556032
test_MSE: 222.88876832252527
```

- 1. The difference between train error and test error of random forest regressor is high, which clearly shows that random forest regressor is overfitting.

Using xgboost

In [6]:

```
import xgboost as xgb
from sklearn.model_selection import GridSearchCV
hyper_parameters = {"max_depth":[1, 2, 3, 4], "n_estimators":[40, 80, 150, 600]}
clf = xgb.XGBRegressor(n_jobs = -1)
best_parameter = GridSearchCV(clf, hyper_parameters, scoring = "neg_mean_absolute_error", cv = 3)
best_parameter.fit(Train_DF, train_TruePickups_flat)
estimators = best_parameter.best_params_["n_estimators"]
depth = best_parameter.best_params_["max_depth"]
```

```
[23:12:21] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:22] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:22] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:22] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:22] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:23] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:23] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:23] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:24] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:24] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:26] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:28] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:30] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:30] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:30] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:31] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:31] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:32] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:32] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:33] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:33] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:34] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:37] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:39] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:42] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:42] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:43] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
```

```
n favor of reg:squarederror.
[23:12:43] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:44] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:44] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:45] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:46] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:47] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:51] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:55] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:58] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:59] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:59] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:12:59] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:13:00] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:13:01] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:13:01] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:13:03] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:13:04] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:13:05] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:13:10] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:13:14] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
[23:13:19] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
```

In [7]:

```
xgb_clf = xgb.XGBRegressor(max_depth = depth, n_estimators = estimators)
xgb_clf.fit(Train_DF, train_TruePickups_flat)
```

```
[23:13:48] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated i
n favor of reg:squarederror.
```

Out[7]:

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0,
             importance_type='gain', learning_rate=0.1, max_delta_step=0,
             max_depth=4, min_child_weight=1, missing=None, n_estimators=80,
             n_jobs=1, nthread=None, objective='reg:linear', random_state=0,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
             silent=None, subsample=1, verbosity=1)
```

In [9]:

```
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
train_pred = xgb_clf.predict(Train_DF)
xgb_train_MAPE = mean_absolute_error(train_TruePickups_flat, train_pred)/
(sum(train_TruePickups_flat)/len(train_TruePickups_flat))
xgb_train_MSE = mean_squared_error(train_TruePickups_flat, train_pred)
```

In [10]:

```
test_pred = xgb_clf.predict(Test_DF)
xgb_test_MAPE = mean_absolute_error(test_TruePickups_flat, test_pred)/ (sum(test_TruePickups_flat)/
len(test_TruePickups_flat))
xgb_test_MSE = mean_squared_error(test_TruePickups_flat, test_pred)
```

In [11]:

```
print(" Using XGBoost model:  ")
print("train_MAPE:",xgb_train_MAPE)
print("train_MSE:",xgb_train_MSE)
print("test_MAPE:", xgb_test_MAPE)
print("test_MSE:", xgb_test_MSE)
```

```
 Using XGBoost model:
train_MAPE: 0.1276545079236112
train_MSE: 260.3512640583455
test_MAPE: 0.11741143576961738
test_MSE: 220.7853657220282
```

Summary

1) First we did some data cleaning on data and then removed outliers from data.

2) After that we built a regression model using some features of frequencies and amplitudes from fast fourier transform of our time series data.

3) And then we applied some regression algorithm such as linear regression, Random Forest Regressor, Xgboost Regressor and compared them using pretty table.

4) In each model we performed hyperparameter tuning using GridSearch, RandomSearch and RandomSearch respectively.

5) After observation we found that the Mean absolute percentage error for Xgboost is better as compared to our linear regression model and Random forest model but in the case of Random forest regressor the difference between train error and test error is very low as compared to other model.

6) The best model with lowest train and test error is XGBoost Regressor.

In [12]:

```
from prettytable import PrettyTable
x = PrettyTable()
x = PrettyTable(["Models", "MAPE train %", "MAPE test %"])
x.add_row(['Linear Regression','13.1','13.2'])
x.add_row(['Random Forest','4.9','11.75'])
x.add_row(['XGBOOST','12.7','11.74'])
print(x)
```

```
+-------------------+--------------+-------------+
|       Models      | MAPE train % | MAPE test % |
+-------------------+--------------+-------------+
| Linear Regression |     13.1     |     13.2    |
|   Random Forest   |     4.9      |    11.75    |
|      XGBOOST      |     12.7     |    11.74    |
+-------------------+--------------+-------------+
```