# TEAM ID: PNT2022TMID43603

# PROJECT NAME: Demand Est - AI powered FoodDemand Forecaster

# Team Leader

0 IBM X ^ Home Page - Select or create a X 2 Code - Jupyter Notebook X + v — O X

C © localhost:8891/notebooks/Downloads/SBSPS-Challenge-8325-Food-Demand-Forecasting-for-Food-Delivery-Company-using-IBM-Cloud-main/SBSPS-Challenge-... \£ 'if CJ ^ :

M Gmail O YouTube if Maps

C jupyter Code (autosave^

^ Logout

jt Trusted | Python 3 (ipykernel)

File Edit View Insert Cell ■nel Widgets Help | C | [[Markdown |

•t«J(i»|♦|ftRim. B |

## Model Evaluation

We're going to use x_train and y_train obtained above in tra i n_test_s pi it section to train our regression model. We're using the fit method and passing the parameters as shown below. Finally we need to check to see how well our model is performing on the test data.

Regression Evaluation Metrics: RMSE:Root Mean Square Error RMSE is the square root of the averaged squared difference between the target value and the value predicted by the model. It is preferred more in some cases because the errors are first squared before averaging which poses a high penalty on large errors. This implies that RMSE is useful when large errors are undesired.

For testing the model we use the below method,

```
In [126]: XG = XGBRegressor()
XG.fit(X_train, y_train) y_pred = XG.predict(X_val) y_pred[y_pred<0] = 0 from sklearn import metrics
print('RMSLE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred))) RMSLE:
70.06429878638917
```

```
In [127]: L = Lasso()
L.fit(X_train, ytrain) y_pred = L.predict(X_val) y_pred[y_pred<0] = 0 from sklearn import metrics
print('RMSLE:100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred))) RMSLE:
128.9558620089095
```

Q IBM X | ^ Home Page - Select or create a X 2 Code - Jupyter Notebook X + v OX

0 © localhost:8891/notebooks/Downloads/SBSPS-Challenge-8325-Food-Demand-Forecasting-for-Food-Delivery-Company-using-IBM-Cloud-main/SBSPS-Challenge-... \£ & d :

M Gmail O YouTube & Maps

3 jupyter Code (autosa»ed)

Logout

| Python 3 (ipykernel)

File Edit View Insert Cell Kernel Widgets Help Not Truss

E1 + *, (h 1b + + ► Run ■ c » Markdown v E3

```
In [128]: EN = ElasticNet()
EN.fit(X_train, y_train) y_pred = EN.predict(X_val) y_pred[y_pred<0] = 0 from sklearn import metrics
print('RMSLE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred))) RMSLE:
130.93230794494932
```

```
In [129]: DT = DecisionTreeRegressor()
DT.fit(X_train, ytrain) y_pred = DT.predict(X_val) y_pred[y_pred<0] = 0 from sklearn import metrics
print('RMSLE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred))) RMSLE:
62.750116693228705
```

```
In [130]: KNN = KNeighborsRegressor()
KNN.fit(X_train, y_train) y_pred = KNN.predict(X_val) y_pred[y_pred<0] = 0 from sklearn import
metrics
print('RMSLE:100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred))) RMSLE:
67.27613082623152
```

```
In [131]: GB = GradientBoostingRegressor() GB.fit(X_train,
y_train)
```

QIBM X | ^ Home Page - Select or create a X Code - Jupyter Notebook X + ˅ O X

0 O localhost:8891/notebooks/Downloads/SBSPS-Challenge-8325-Food-Demand-Forecasting-for-Food-Delivery-Company-using-IBM-Cloud-main/SBSPS-Challenge-... \£ 'tl Q :

M Gmail O YouTube IJf Maps

^ jupyter CodG (autosaved) f® Logout

I Not Trusted | Python 3 (ipykemel) *o

File Edit View Insert Cell Kernel Widgets Help El + 3 Q C ♦ + ► Run ■ C ►►

Markdown v o

RMSLE: 130.93230794494932

```
In [129]: DT = DecisionTreeRegressorQ DT.fit(X_train, y_train) y_pred = DT.predict(X_val) y_pred[y_pred<0] = 0 from sklearn
          import metrics

print('RMSLE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred))) RMSLE: 62.750116693228705
In [130]: KNN = KNeighborsRegressor()
KNN.fit(X_train, y train) y_pred = KNN.predict(X_val) y_pred[y_pred<0] = 0 from sklearn import metrics
print('RMSLE: 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred))) RMSLE: 67.27613082623152
```

```
In [131]: GB = GradientBoostingRegressorQ
GB.fit(X_train, y_train) y_pred = GB.predict(X_val)
y_pred[y_pred<0] = 0 from sklearn import metrics

print('RMSLE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred))) RMSLE:

99.04866931366767
```

# Team Member 1

## Model Evaluation

We're going to use x_train and y_train obtained above in tra i n_test_s pi it section to train our regression model. Were using the fit method and passing the parameters as shown below. Finally, we need to check to see how well our model is performing on the test data.

Regression Evaluation Metrics: RMSE:Root Mean Square Error RMSE is the square root of the averaged squared difference between the target value and the value predicted by the model. It is preferred more in some cases because the errors are first squared before averaging which poses a high penalty on large errors. This implies that RMSE is useful when large errors are undesired.

For testing the model we use the below method,

In [126]:
```
XG - XGBRegressorQ
XG.fit(X_train, y_train) y_pred = XG.predict(X_val) y_pred[y_pred<0] = 0 from sklearn Import metrics
print('RMSLE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))
```

RMSLE: 70.06429878638917

In [127]:
```
L = Lasso()
L.fit(X_train, ytrain) y_pred = l.predict(X_val) y_pred[y_pred<0] = 0 from sklearn import metrics
print('RMSLE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))
```

RMSLE: 128.9558620089095



In [128]:
```
EN = ElasticNet()
EN.fit(X_train, y_train) y_pred =
EN.predict(X_val) y_pred[y_pred<0]
= 0 from sklearn import metrics

print('RMSLE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))
```

RMSLE: 130.93230794494932

In [129]:
```
DT = DecisionTreeRegressorQ DT.fit(X_train, ytrain) y_pred = DT.predict(X_val) y_pred[y_pred<0] = 0
from sklearn import metrics
print('RMSLE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))
```

RMSLE: 62.750116693228705

```
KNN = KNeighborsRegressor()
KNN.fit(X_train, y_train) y_pred = KNN.predict(X_val) y_pred[y_pred<0] = 0 from sklearn import
metrics
print('RMSLE:100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))
```

RMSLE: 67.27613082623152

In [131]:
```
GB = GradientBoostingRegressorQ
GB.fit(X_train, y_train)
```

RMSLE: 130.93230794494932

In [129]: DT = DecisionTreeRegressorQ DT.fit(X_train, y_train) y_pred = DT.predict(X_val) y_pred[y_pred<0] = 0 from sklearn import metrics

print('RMSLE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred))) RMSLE: 62.750116693228705

In [130]: KNN = KNeighborsRegressor()
KNN.fit(X_train, ytrain) y_pred = KNN.predict(X_val)y_pred[y_pred<0] = 0 from sklearn import metrics

print('RMSLE: 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred))) RMSLE: 67.27613082623152

In [131]: GB = GradientBoostingRegressor()
GB.fit(X_train, y_train) y_pred =
GB.predict(X_val) y_pred[y_pred<0]
= 0 from sklearn import metrics

print('RMSLE:100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred))) RMSLE:

99.04866931366767