

TEAM ID: PNT2022TMID17645

## PROJECT NAME: DemandEst - AI powered Food Demand Forecaster

### Team Leader

The screenshot displays a Jupyter Notebook interface with a browser window at the top showing the URL `localhost:8891/notebooks/Downloads/SBSPS-Challenge-8325-Food-Demand-Forecasting-for-Food-Delivery-Company-using-IBM-Cloud-main/SBSPS-Challenge-...`. The notebook is titled "jupyter Code (autosaved)" and shows a "Not Connected" status for the kernel. The content of the notebook is as follows:

### Model Evaluation

We're going to use `x_train` and `y_train` obtained above in `train_test_split` 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('RMSE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))

RMSE: 70.06429878638917
```

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

RMSE: 128.9558620889095
```

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

RMSE: 130.93230794494932
```

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

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

RMSE: 67.27613082623152
```

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

```
RMSLE: 130.93230794494932

In [129]: DT = DecisionTreeRegressor()
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 = 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
```

## Team Member 1

### Model Evaluation

We're going to use `x_train` and `y_train` obtained above in `train_test_split` 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, y_train)
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
```

IBM x Home Page - Select or create a n x Code - Jupyter Notebook x +

localhost:8891/notebooks/Downloads/SBSPS-Challenge-8325-Food-Demand-Forecasting-for-Food-Delivery-Company-using-IBM-Cloud-main/SBSPS-Challenge-... ☆ □

Gmail YouTube Maps

jupyter Code (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Connected Not Trusted Python 3 (pykernel)

Run

```
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, 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 = GradientBoostingRegressor()
          GB.fit(X_train, y_train)
```

IBM x Home Page - Select or create a n x Code - Jupyter Notebook x +

localhost:8891/notebooks/Downloads/SBSPS-Challenge-8325-Food-Demand-Forecasting-for-Food-Delivery-Company-using-IBM-Cloud-main/SBSPS-Challenge-... ☆ □

Gmail YouTube Maps

jupyter Code (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Connected Not Trusted Python 3 (pykernel)

Run

```
RMSLE: 130.93230794494932
```

```
In [129]: DT = DecisionTreeRegressor()
          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
```

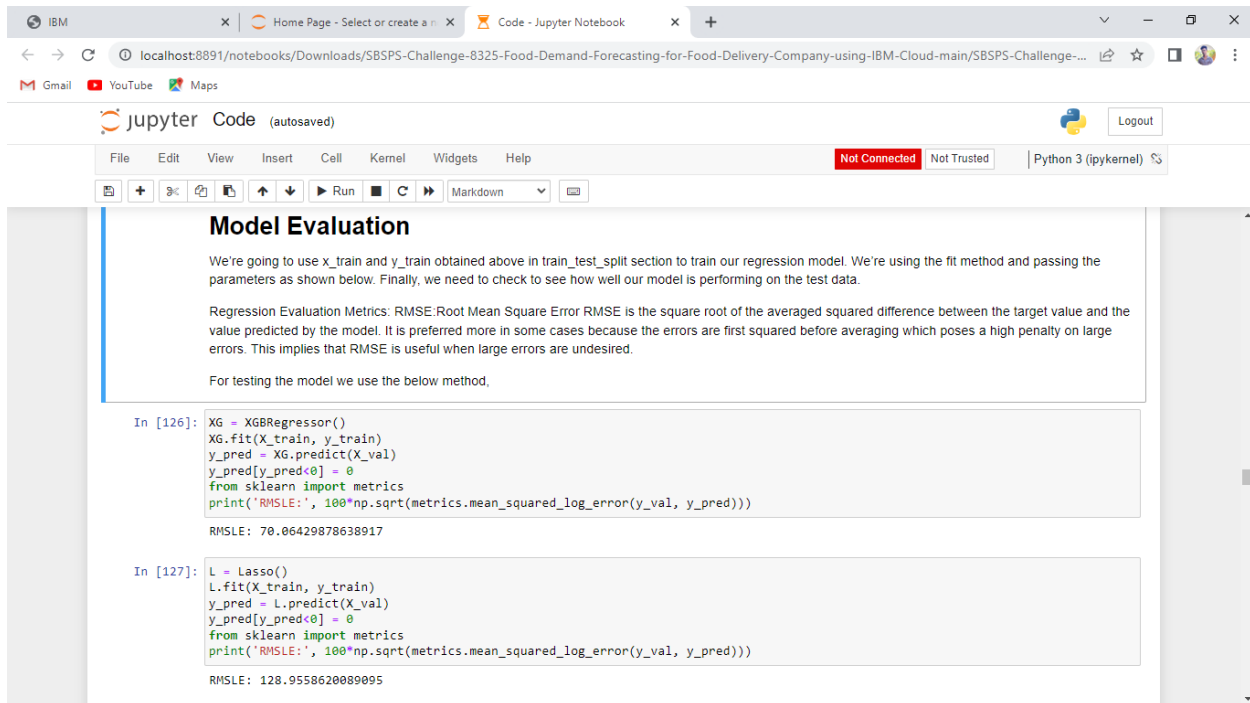
```
In [130]: KNN = KNeighborsRegressor()
          KNN.fit(X_train, y_train)
          y_pred = KNN.predict(X_val)
          y_pred[y_pred<0] = 0
          from sklearn import metrics
          print('RMSLE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))

RMSLE: 67.27613082623152
```

```
In [131]: GB = 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
```

## Team Member 2



The screenshot shows a Jupyter Notebook interface with a browser window at the top. The notebook is titled "jupyter Code (autosaved)" and has a menu bar with File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The status bar indicates "Not Connected", "Not Trusted", and "Python 3 (ipykernel)".

### Model Evaluation

We're going to use `x_train` and `y_train` obtained above in `train_test_split` 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.

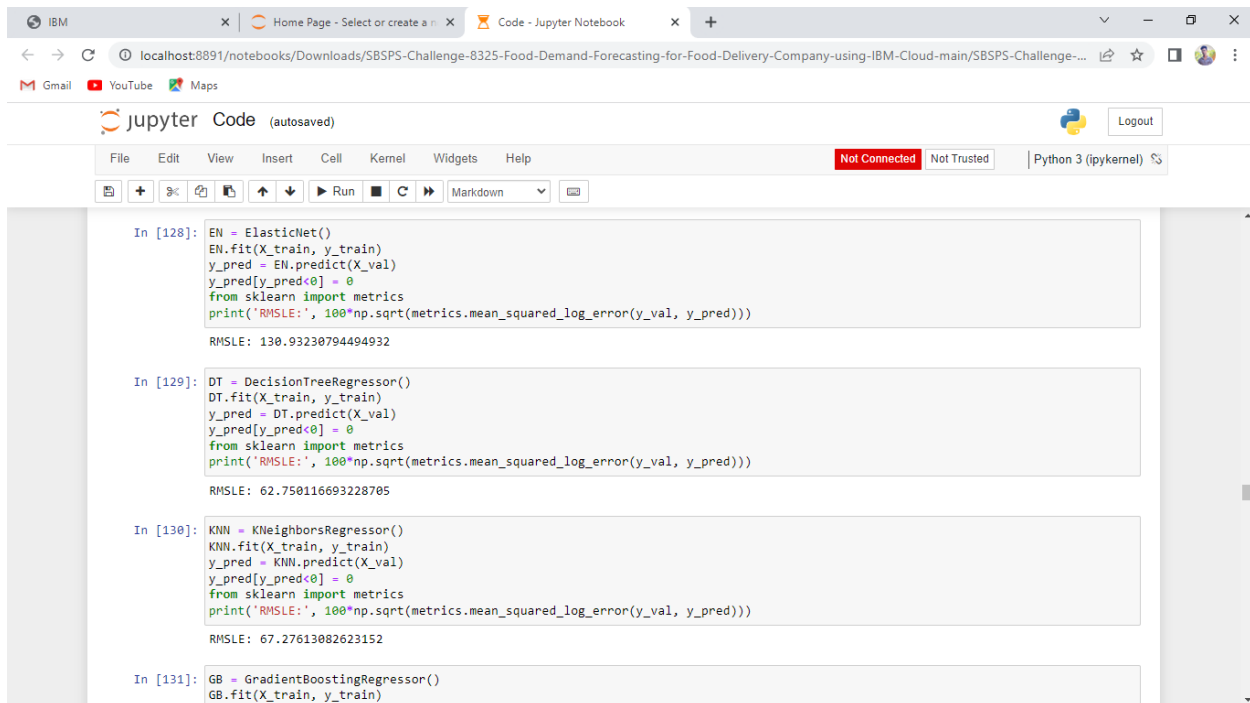
For testing the model we use the below method,

```
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, y_train)
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
```



The screenshot shows a Jupyter Notebook interface with a browser window at the top. The notebook is titled "jupyter Code (autosaved)" and has a menu bar with File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The status bar indicates "Not Connected", "Not Trusted", and "Python 3 (ipykernel)".

```
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, 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 = GradientBoostingRegressor()
GB.fit(X_train, y_train)
```

```
RMSLE: 130.93230794494932

In [129]: DT = DecisionTreeRegressor()
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 = 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
```

## Team Member 3

### Model Evaluation

We're going to use `x_train` and `y_train` obtained above in `train_test_split` 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, y_train)
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
```

IBM x Home Page - Select or create a n x Code - Jupyter Notebook x +

localhost:8891/notebooks/Downloads/SBSPS-Challenge-8325-Food-Demand-Forecasting-for-Food-Delivery-Company-using-IBM-Cloud-main/SBSPS-Challenge-...

Gmail YouTube Maps

jupyter Code (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Connected Not Trusted Python 3 (ipykernel)

Run

```
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, 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 = GradientBoostingRegressor()
          GB.fit(X_train, y_train)
```

IBM x Home Page - Select or create a n x Code - Jupyter Notebook x +

localhost:8891/notebooks/Downloads/SBSPS-Challenge-8325-Food-Demand-Forecasting-for-Food-Delivery-Company-using-IBM-Cloud-main/SBSPS-Challenge-...

Gmail YouTube Maps

jupyter Code (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Connected Not Trusted Python 3 (ipykernel)

Run

```
RMSLE: 130.93230794494932
```

```
In [129]: DT = DecisionTreeRegressor()
          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 = 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
```