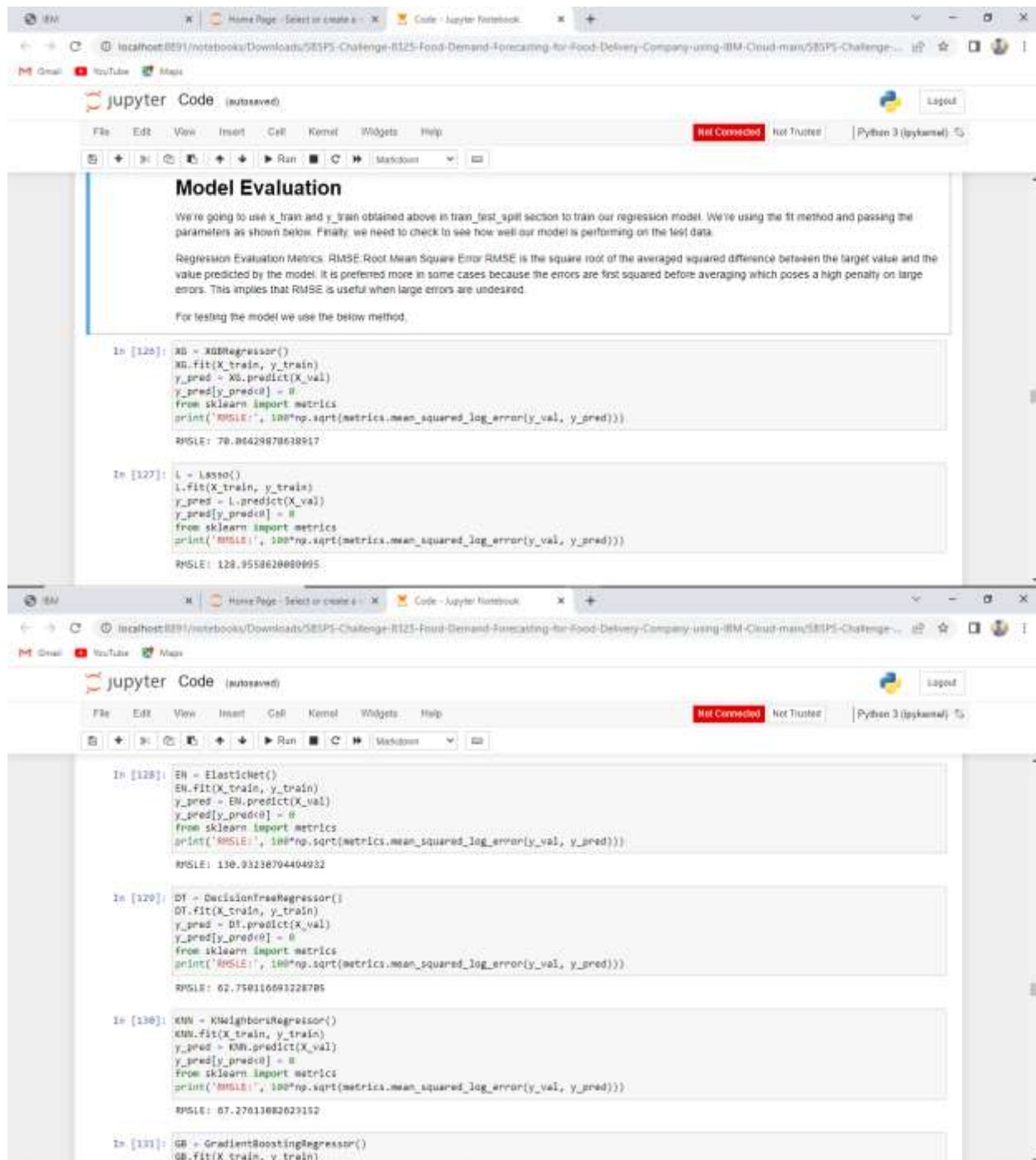


TEAM ID: PNT2022TMID32368

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/SESPS-Challenge-R125-Food-Demand-Forecasting-for-Food-Delivery-Company-using-IBM-Cloud-man/SESPS-Challenge-... The notebook is titled "Model Evaluation" and contains the following text:

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.

The notebook contains four code cells, each evaluating a different regression model and printing the RMSE:

```
In [125]:
XB = XGBRegressor()
XB.fit(X_train, y_train)
y_pred = XB.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: 70.06429870638917
```

```
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: 128.9558610080995
```

```
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: 130.93138704404032
```

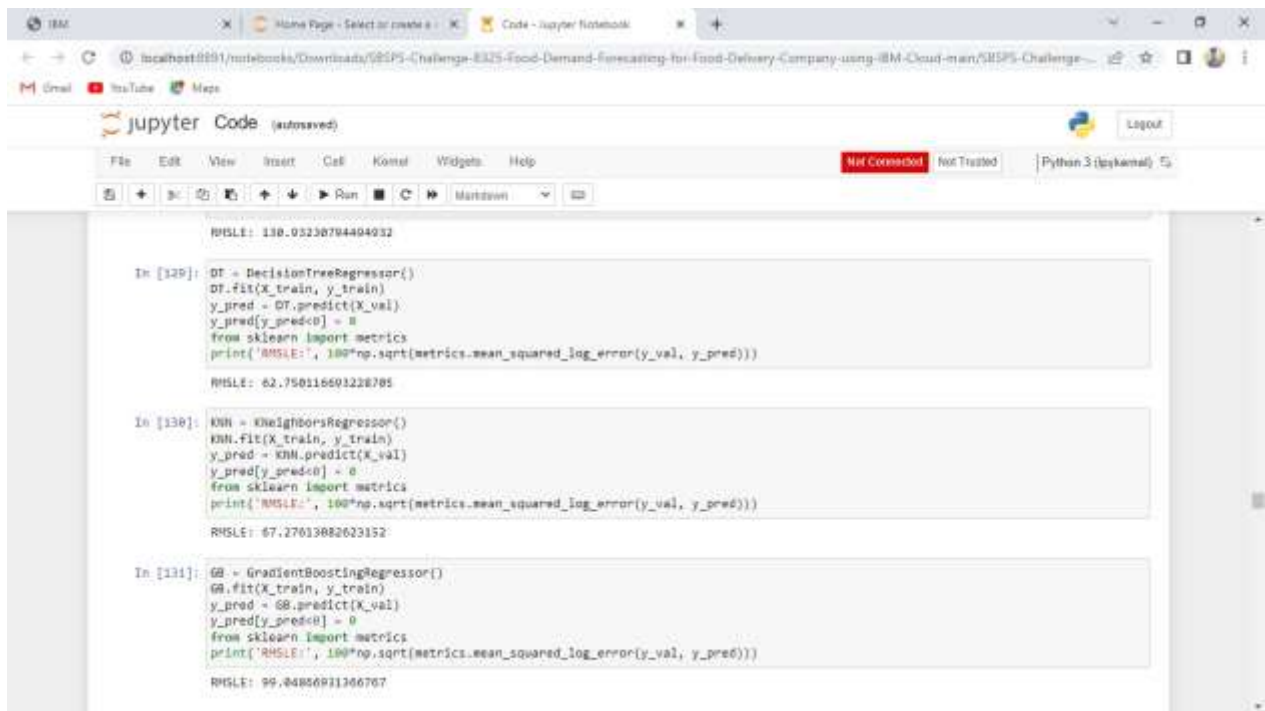
```
In [129]:
DF = DecisionTreeRegressor()
DF.fit(X_train, y_train)
y_pred = DF.predict(X_val)
y_pred[y_pred < 0] = 0
from sklearn import metrics
print('RMSE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))

RMSE: 62.750116693228795
```

```
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)
```



The screenshot shows a Jupyter Notebook interface with three code cells. Each cell calculates the Root Mean Square Error (RMSE) for a different regression model. The first cell uses a DecisionTreeRegressor, the second uses a KNeighborsRegressor, and the third uses a GradientBoostingRegressor. Each cell prints the RMSE value at the end of the execution.

```
RMSLE: 138.95238794494932
```

```
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: 62.750116603228785
```

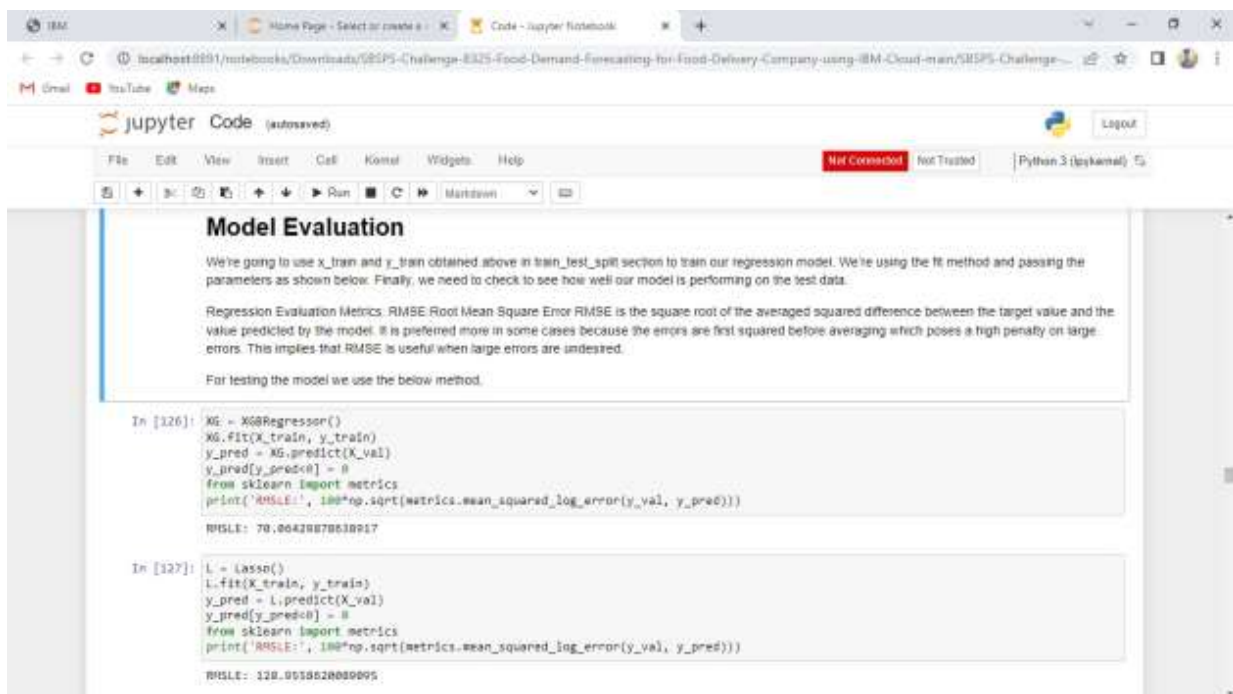
```
In [130]: KNN = KNeighborsRegressor()
KNN.fit(X_train, y_train)
y_pred = KNN.predict(X_val)
y_pred[y_pred<0] = 0
from sklearn import metrics
print('RMSLE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))

RMSLE: 67.27013882623152
```

```
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: 99.84856911366767
```

Team Member 1



The screenshot shows a Jupyter Notebook interface. The first cell contains a text block titled "Model Evaluation" which explains the use of x\_train and y\_train for training and the importance of checking model performance on test data. It also defines RMSE (Root Mean Square Error) and provides a formula for its calculation. The second cell calculates the RMSE for an XGBRegressor, and the third cell calculates the RMSE for a Lasso model.

### 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: 70.86428878538917
```

```
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: 128.80196288089095
```

IBM Home Page - Select or create a workspace Code - Jupyter Notebook

localhost:8891/notebooks/Downloads/S&SPS-Challenge-E&ZS-Food-Demand-Forecasting-for-Food-Delivery-Company-using-IBM-Cloud-man/S&SPS-Challenge...

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

RMSE: 138.93238794494932
```

```
In [129]: DF = DecisionTreeRegressor()
DF.fit(X_train, y_train)
y_pred = DF.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: 62.758116693228785
```

```
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: 67.27613882623152
```

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

IBM Home Page - Select or create a workspace Code - Jupyter Notebook

localhost:8891/notebooks/Downloads/S&SPS-Challenge-E&ZS-Food-Demand-Forecasting-for-Food-Delivery-Company-using-IBM-Cloud-man/S&SPS-Challenge...

jupyter Code (autosaved) Logout

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

Run

```
RMSE: 138.93238794494932
```

```
In [129]: DF = DecisionTreeRegressor()
DF.fit(X_train, y_train)
y_pred = DF.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: 62.758116693228785
```

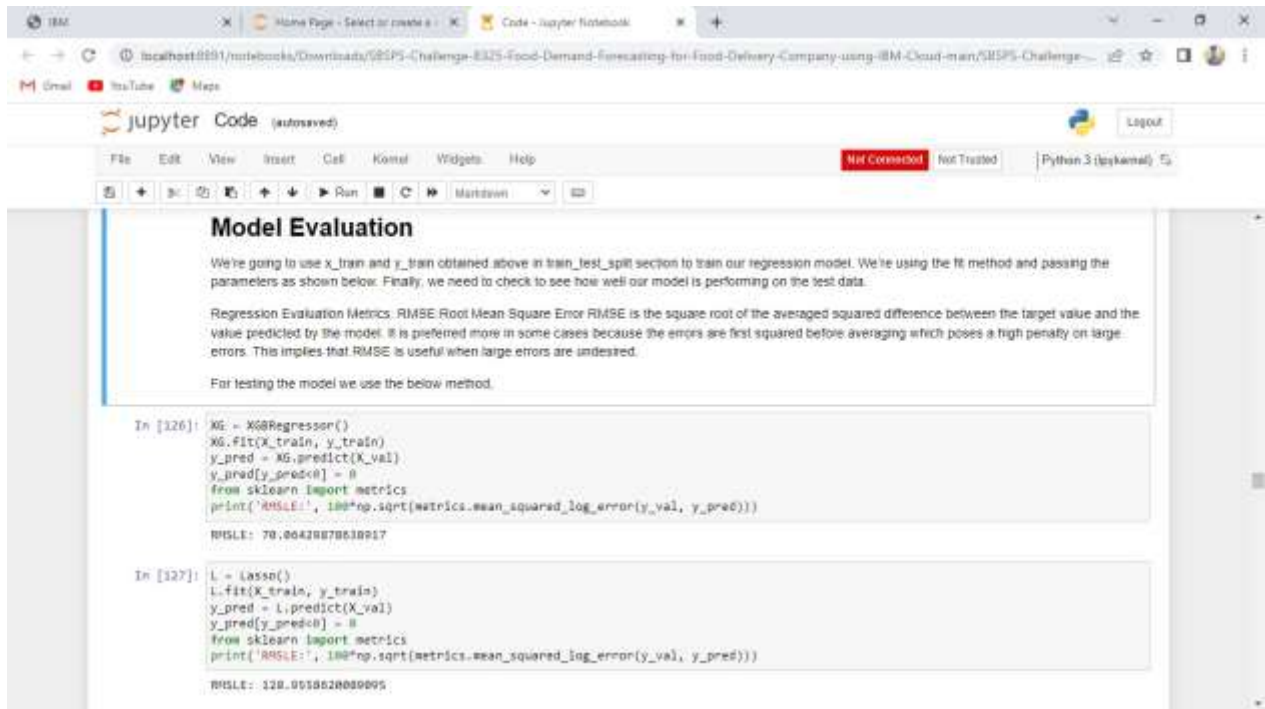
```
In [130]: KNN = KNeighborsRegressor()
KNN.fit(X_train, y_train)
y_pred = KNN.predict(X_val)
y_pred[y_pred<0] = 0
from sklearn import metrics
print('RMSE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))

RMSE: 67.27613882623152
```

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

RMSE: 99.04856933366767
```

## Team Member 2



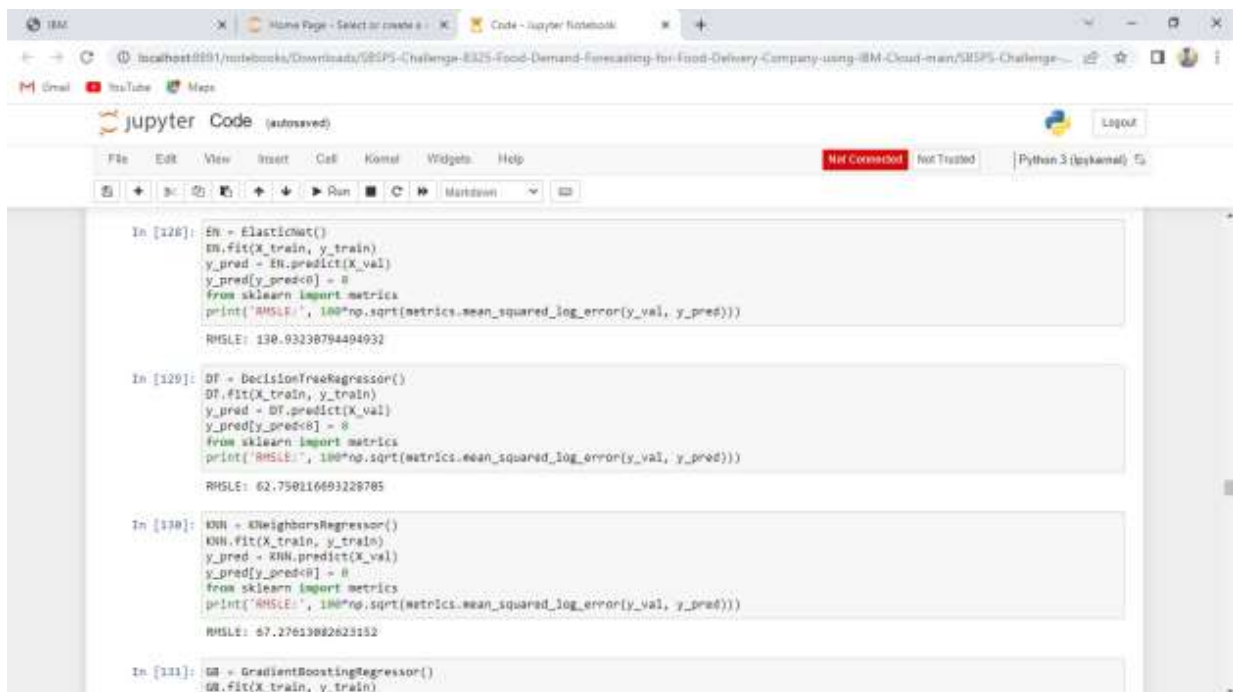
The screenshot shows a Jupyter Notebook interface with a browser window at the top. The notebook is titled "Model Evaluation" and contains two code cells. The first cell, labeled "In [126]", defines an XGBRegressor, fits it to training data, predicts on validation data, and prints the RMSE. The second cell, labeled "In [127]", defines a Lasso regressor, fits it to training data, predicts on validation data, and prints the RMSE. The notebook status bar at the bottom indicates "Not Connected" and "Not Trusted".

```
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: 70.00428878538917

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: 128.9518628089895
```



The screenshot shows a Jupyter Notebook interface with a browser window at the top. The notebook contains four code cells, each testing a different regression model. The first cell, labeled "In [128]", tests an ElasticNet regressor. The second cell, labeled "In [129]", tests a DecisionTree regressor. The third cell, labeled "In [130]", tests a KNeighbors regressor. The fourth cell, labeled "In [131]", tests a GradientBoosting regressor. Each cell prints the RMSE. The notebook status bar at the bottom indicates "Not Connected" and "Not Trusted".

```
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: 130.93238794494932

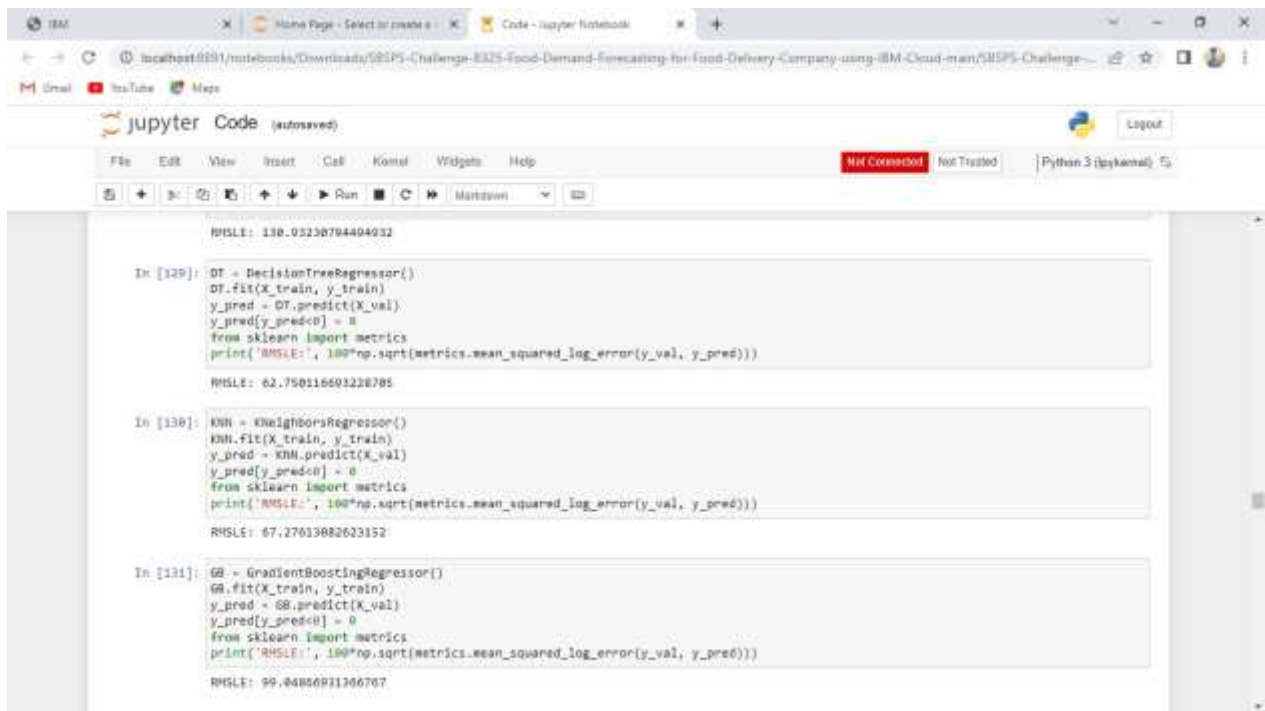
In [129]: DT = DecisionTreeRegressor()
DT.fit(X_train, y_train)
y_pred = DT.predict(X_val)
y_pred[y_pred<0] = 0
from sklearn import metrics
print('RMSE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))

RMSE: 62.750116693226785

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: 67.27613882823152

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



```
RMSE: 138.95238794494952

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: 62.750116603228785

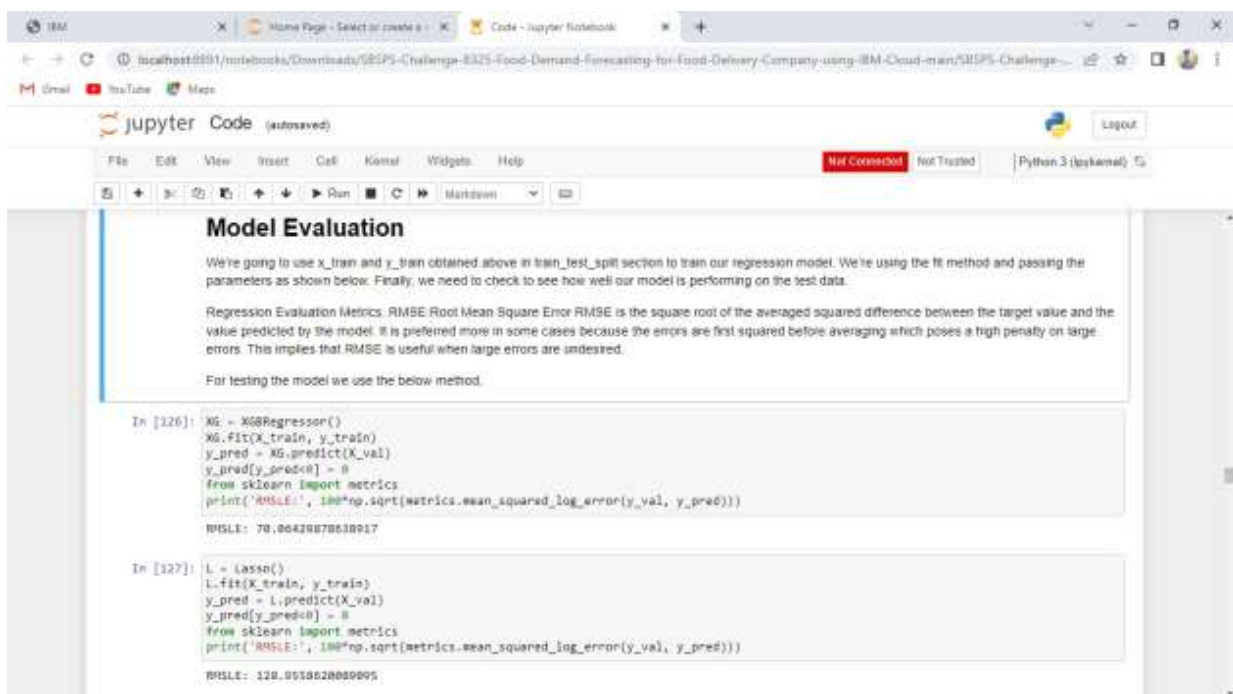
In [130]: KNN = KNeighborsRegressor()
KNN.fit(X_train, y_train)
y_pred = KNN.predict(X_val)
y_pred[y_pred<0] = 0
from sklearn import metrics
print('RMSE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))

RMSE: 67.27013882623152

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

RMSE: 99.84856931366767
```

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

RMSE: 70.00428878638917

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: 128.95186288889995
```



IBM Home Page - Select or create a workspace Code - Jupyter Notebook

localhost:8891/notebooks/Downloads/SEPS-Challenge-E325-Food-Demand-Forecasting-for-Food-Delivery-Company-using-IBM-Cloud-man/SEPS-Challenge...

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('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: 62.750116693228785
```

```
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)
```

IBM Home Page - Select or create a workspace Code - Jupyter Notebook

localhost:8891/notebooks/Downloads/SEPS-Challenge-E325-Food-Demand-Forecasting-for-Food-Delivery-Company-using-IBM-Cloud-man/SEPS-Challenge...

jupyter Code (autosaved) Logout

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

Run

```
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: 62.750116693228785
```

```
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)
y_pred = GB.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: 99.04856911366767
```