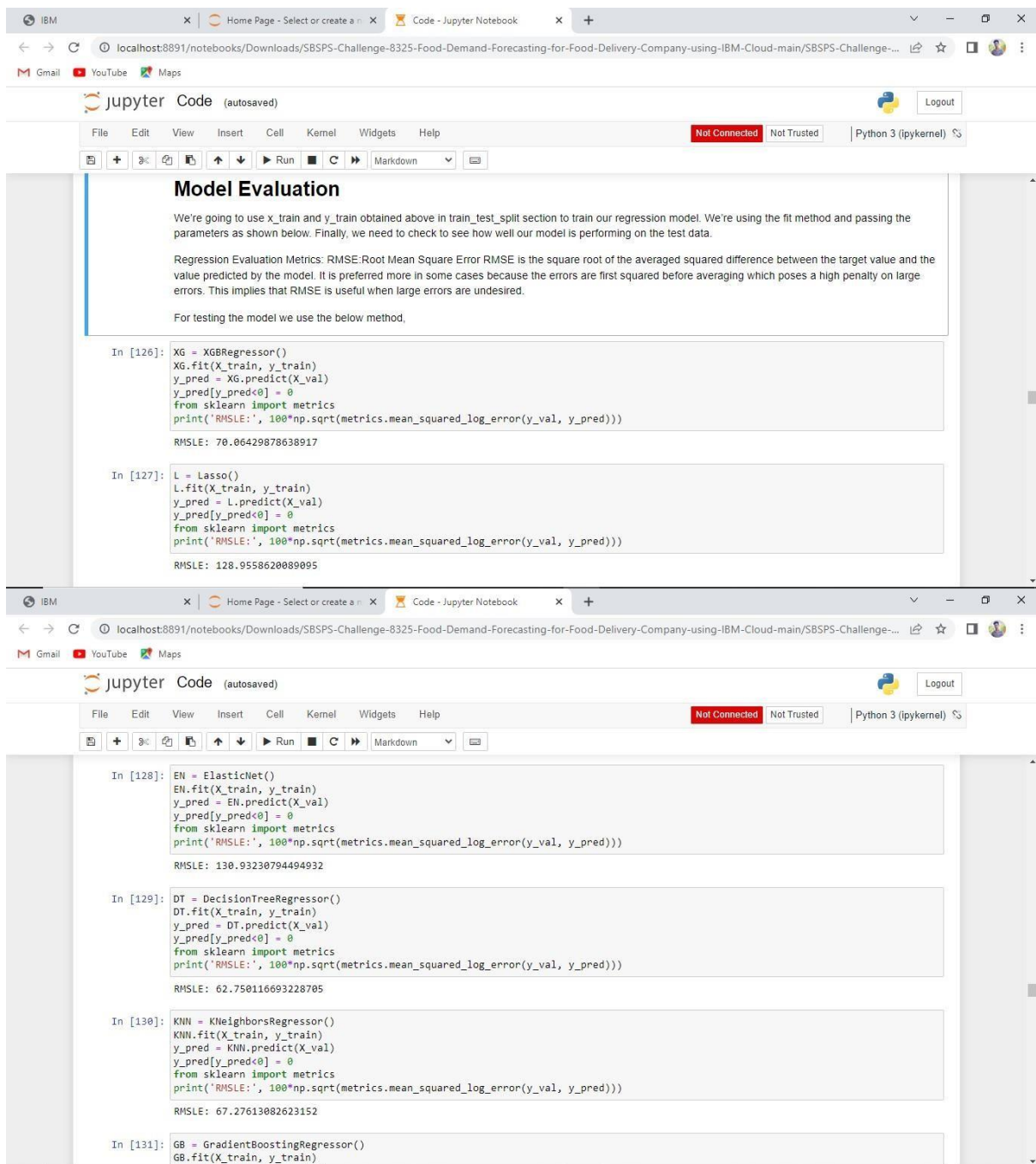


TEAM ID: PNT2022TMID14821

PROJECT NAME: DemandEst - AI powered Food Demand Forecaster

Team Leader



The screenshot displays a Jupyter Notebook interface with the following content:

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: 128.9558620089095
```

```
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('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)
```

The screenshot shows a Jupyter Notebook interface with three code cells. The first cell uses DecisionTreeRegressor and outputs an RMSLE of 130.93230794494932. The second cell uses KNeighborsRegressor and outputs an RMSLE of 62.750116693228705. The third cell uses GradientBoostingRegressor and outputs an RMSLE of 99.04866931366767. The interface includes a browser window at the top, a Jupyter logo, and a 'Not Connected' status indicator.

```
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

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

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

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: 128.9558620009095
```

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

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

RMSE: 130.93230794494932
```

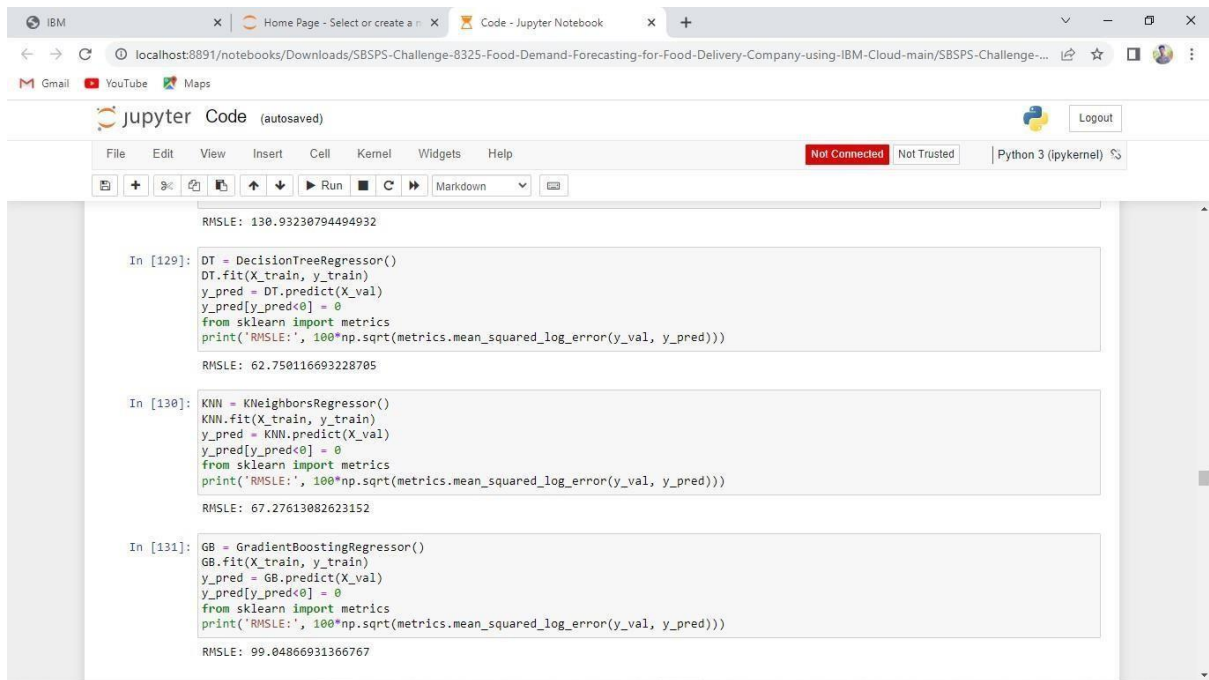
```
In [129]: DT = DecisionTreeRegressor()
DT.fit(X_train, y_train)
y_pred = DT.predict(X_val)
y_pred[y_pred<0] = 0
from sklearn import metrics
print('RMSE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))

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)
```



The screenshot shows a Jupyter Notebook interface with three code cells. Each cell contains Python code for training a model and calculating the Root Mean Squared Logarithmic Error (RMSLE). The output of each cell is the RMSLE value.

```
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

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

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

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: 128.9558620009095
```

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

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

RMSE: 130.93230794494932
```

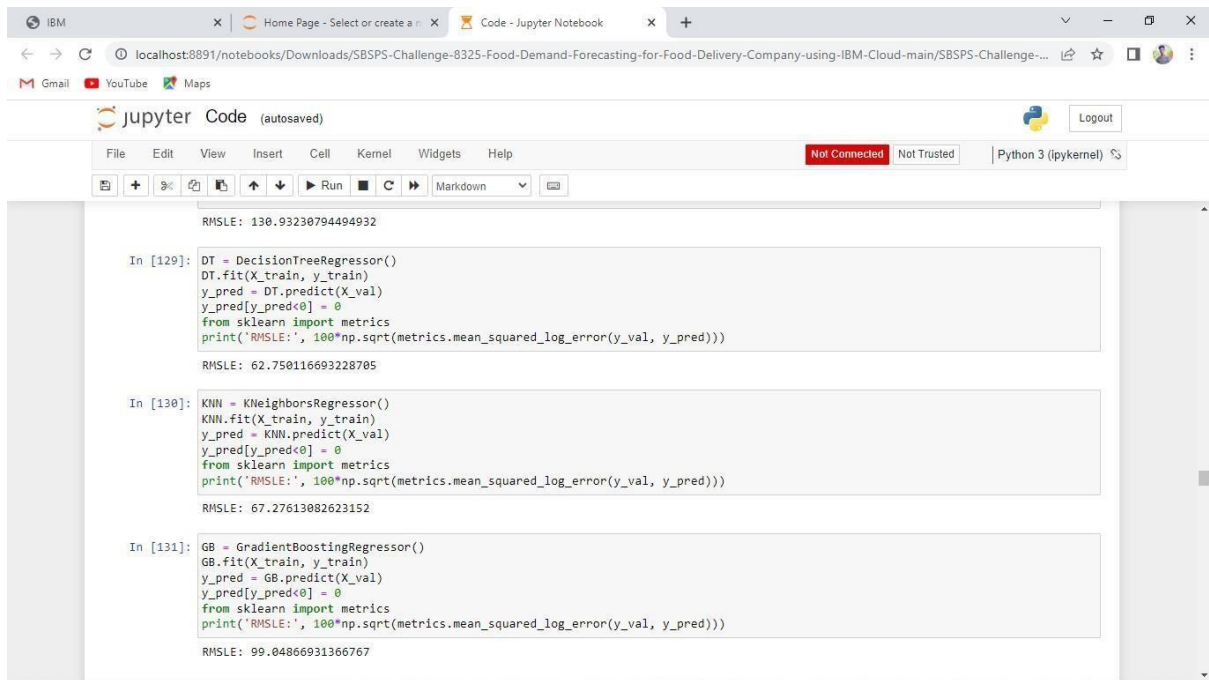
```
In [129]: DT = DecisionTreeRegressor()
DT.fit(X_train, y_train)
y_pred = DT.predict(X_val)
y_pred[y_pred<0] = 0
from sklearn import metrics
print('RMSE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))

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)
```



The screenshot shows a Jupyter Notebook interface with three code cells. Each cell contains Python code for a different machine learning model and its corresponding RMSLE value.

```
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

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

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

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: 128.9558620009095
```

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

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('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)
```

IBM | Home Page - Select or create a notebook | Code - Jupyter Notebook

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 | Markdown

```
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
```