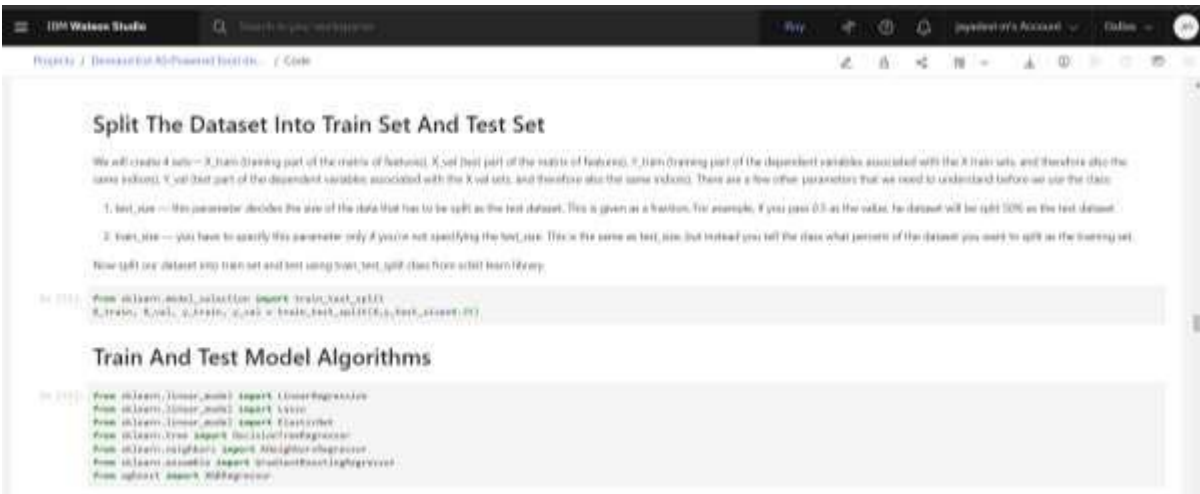


# SPRINT-3

## MODEL BUILDING

|               |   |
|---------------|---|
| Team ID       | PNT2022TMID46533                            |
| Project Title | DemandEst-AI Powered Food Demand Forecaster |

### Train and test model algorithms



The screenshot shows the IBM Watson Studio interface. The main content area displays a Jupyter Notebook with the following sections:

#### Split The Dataset Into Train Set And Test Set

We will create 4 sets —  $X_{train}$  (training part of the matrix of features),  $X_{test}$  (test part of the matrix of features),  $y_{train}$  (training part of the dependent variables associated with the  $X_{train}$  sets, and therefore also the same index),  $y_{test}$  (test part of the dependent variables associated with the  $X_{test}$  sets, and therefore also the same index). There are a few other parameters that we need to understand before we use the data:

1. `test_size` — this parameter decides the size of the data that has to be split as the test dataset. This is given as a fraction. For example, if you pass 0.5 as the value, the dataset will be split 50% as the test dataset.
2. `train_size` — you have to specify this parameter only if you're not specifying the `test_size`. This is the same as `test_size`, but instead you tell the class what percent of the dataset you want to split as the training set.


Now split our dataset into train set and test using `train_test_split` from `sklearn`:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

#### Train And Test Model Algorithms

```
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import mean_squared_error
```

### Model Evaluation



The screenshot shows the IBM Watson Studio interface. The main content area displays a Jupyter Notebook with the following sections:

#### 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. Firstly, we need to check to see how well our model is performing on the test data.

Regression Evaluation Metric: 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 gives a high penalty on large errors. This implies that RMSE is useful when large errors are undesirable.

For testing the model we use the below method:

```
RM = RidgeRegressor()
RM.fit(X_train, y_train)
y_pred = RM.predict(X_test)
y_grad(y_pred) = 0
from sklearn.metrics import mean_squared_error
print('RMSE: ', 100*sq(mean_squared_error(y_test, y_pred)))
RMSE: 55.107134779889
```

```
L = Logistic()
L.fit(X_train, y_train)
y_pred = L.predict(X_test)
y_grad(y_pred) = 0
from sklearn.metrics import mean_squared_error
print('RMSE: ', 100*sq(mean_squared_error(y_test, y_pred)))
RMSE: 338.508854884489
```

```
RM = RandomForest()
RM.fit(X_train, y_train)
y_pred = RM.predict(X_test)
y_grad(y_pred) = 0
from sklearn.metrics import mean_squared_error
print('RMSE: ', 100*sq(mean_squared_error(y_test, y_pred)))
```









