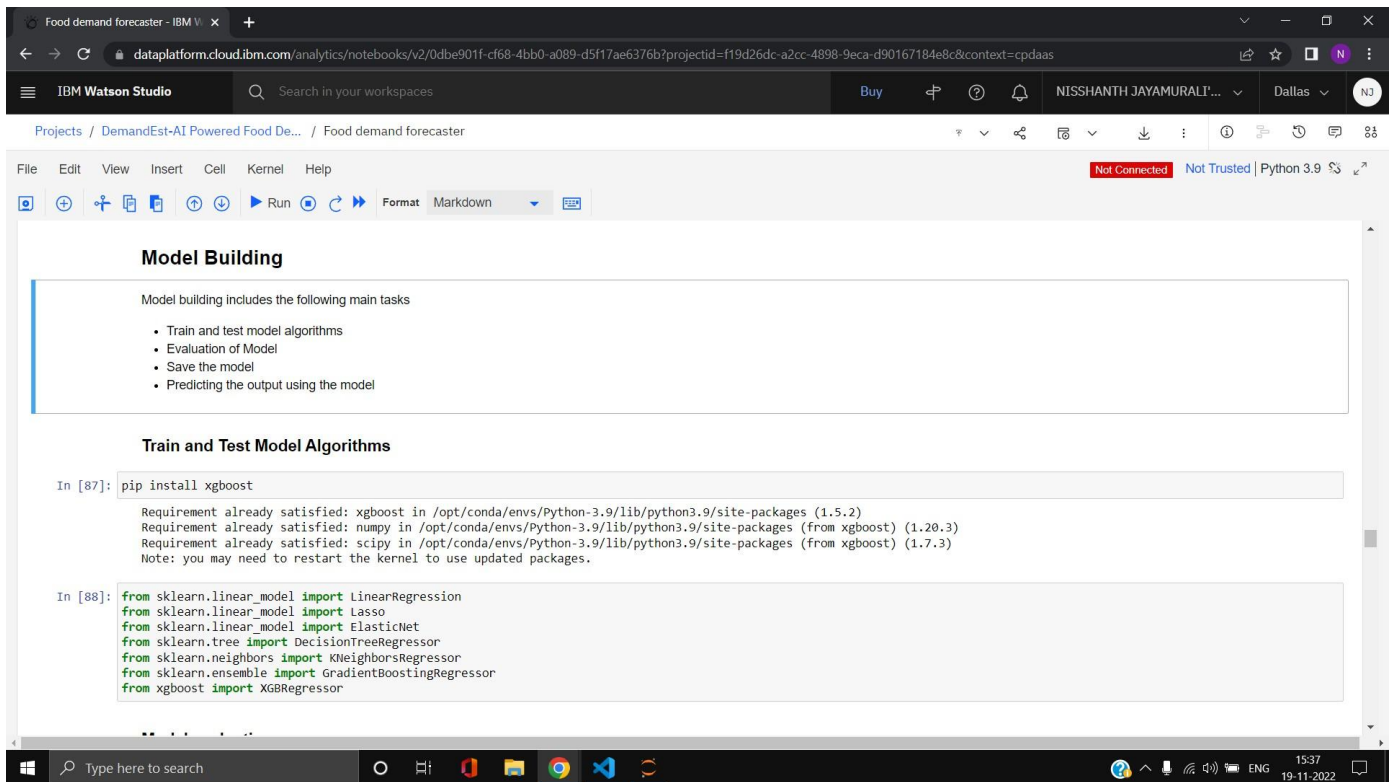


TRAIN THE ML MODEL ON IBM

Team id: PNT2022TMID15280

TRAIN AND TEST MODEL ALGORITHMS



Food demand forecaster - IBM V... x +

datapatform.cloud.ibm.com/analytics/notebooks/v2/0dbe901f-cf68-4bb0-a089-d5f17ae6376b?projectid=f19d26dc-a2cc-4898-9eca-d90167184e8c&context=cpdaas

IBM Watson Studio Search in your workspaces Buy

Projects / DemandEst-AI Powered Food De... / Food demand forecaster

File Edit View Insert Cell Kernel Help Not Connected Not Trusted Python 3.9

Model Building

Model building includes the following main tasks

- Train and test model algorithms
- Evaluation of Model
- Save the model
- Predicting the output using the model

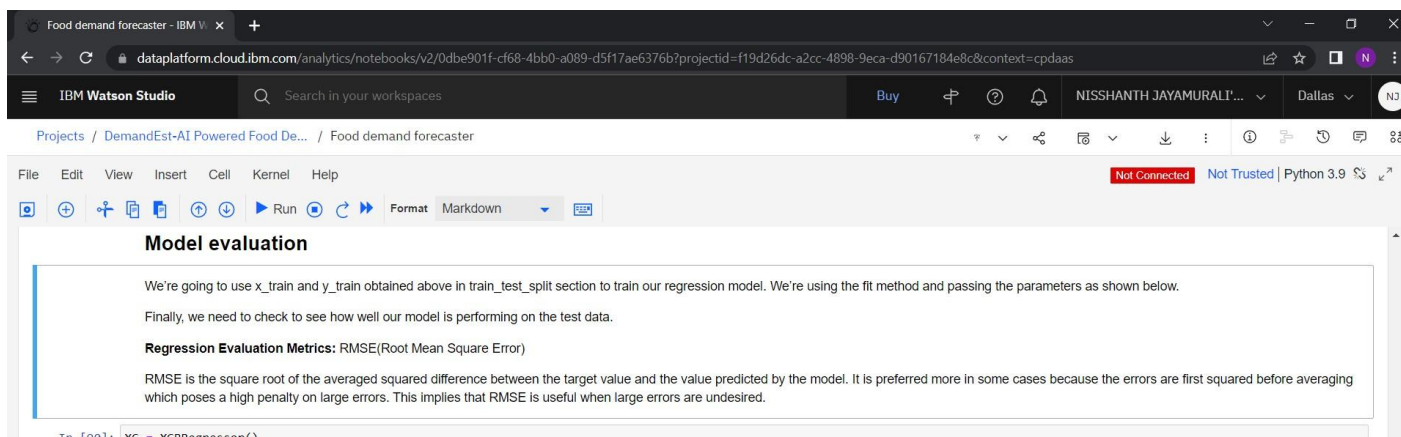
Train and Test Model Algorithms

In [87]: `pip install xgboost`

Requirement already satisfied: xgboost in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (1.5.2)
Requirement already satisfied: numpy in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from xgboost) (1.20.3)
Requirement already satisfied: scipy in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from xgboost) (1.7.3)
Note: you may need to restart the kernel to use updated packages.

In [88]: `from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import GradientBoostingRegressor
from xgboost import XGBRegressor`

MODEL EVALUATION



Food demand forecaster - IBM V... x +

datapatform.cloud.ibm.com/analytics/notebooks/v2/0dbe901f-cf68-4bb0-a089-d5f17ae6376b?projectid=f19d26dc-a2cc-4898-9eca-d90167184e8c&context=cpdaas

IBM Watson Studio Search in your workspaces Buy

Projects / DemandEst-AI Powered Food De... / Food demand forecaster

File Edit View Insert Cell Kernel Help Not Connected Not Trusted Python 3.9

Model evaluation

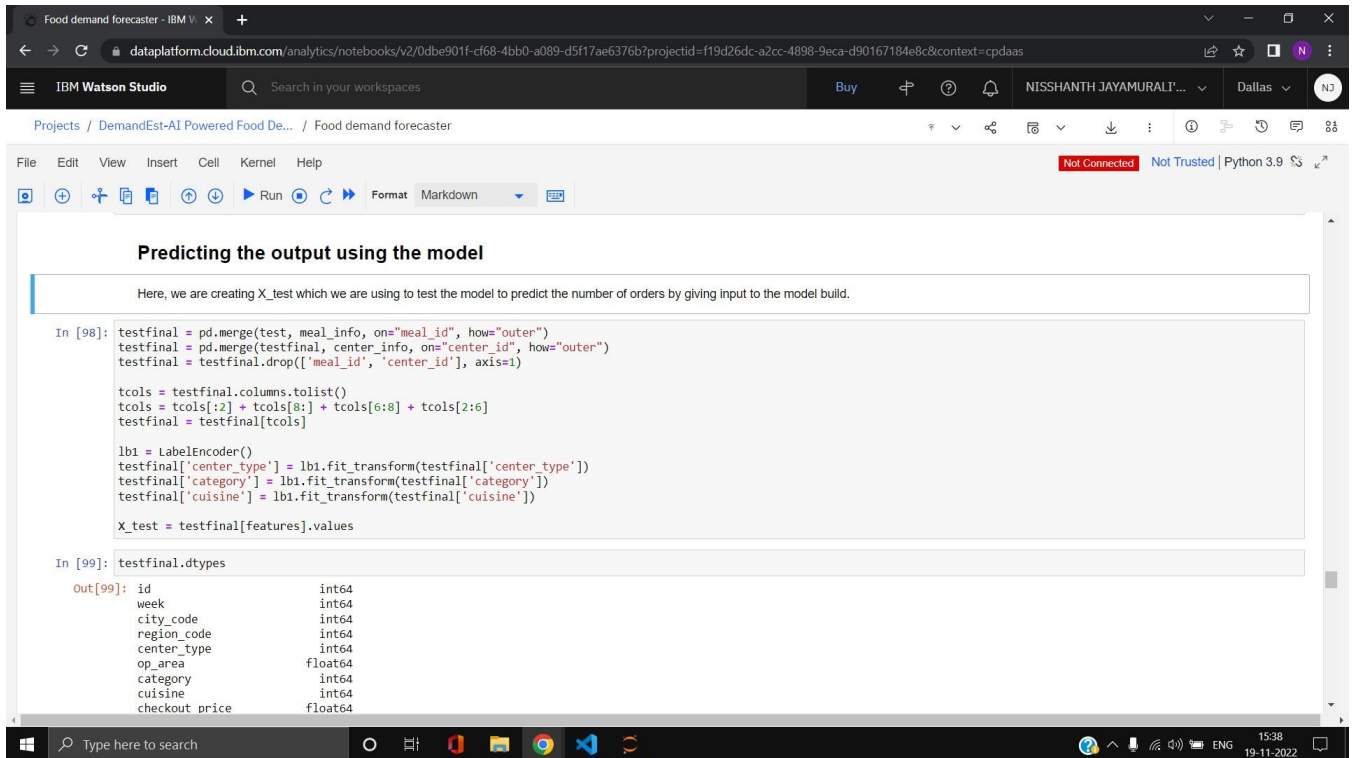
We're going to use `x_train` and `y_train` obtained above in `train_test_split` section to train our regression model. We're using the `fit` method and passing the parameters as shown below.

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.

PREDICTING THE OUTPUT USING THE MODEL



Predicting the output using the model

Here, we are creating `X_test` which we are using to test the model to predict the number of orders by giving input to the model build.

```
In [98]: testfinal = pd.merge(test, meal_info, on="meal_id", how="outer")
testfinal = pd.merge(testfinal, center_info, on="center_id", how="outer")
testfinal = testfinal.drop(['meal_id', 'center_id'], axis=1)

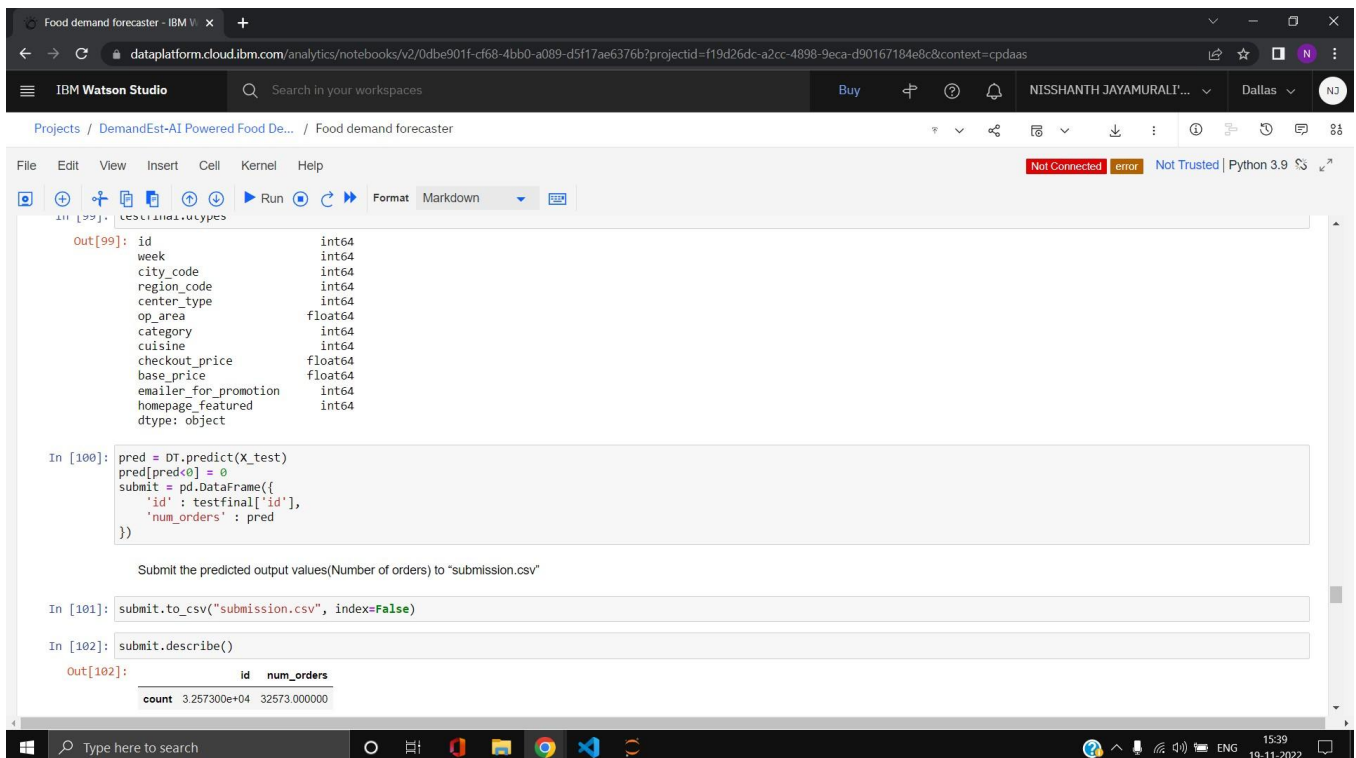
tcols = testfinal.columns.tolist()
tcols = tcols[:2] + tcols[8:] + tcols[6:8] + tcols[2:6]
testfinal = testfinal[tcols]

lb1 = LabelEncoder()
testfinal['center_type'] = lb1.fit_transform(testfinal['center_type'])
testfinal['category'] = lb1.fit_transform(testfinal['category'])
testfinal['cuisine'] = lb1.fit_transform(testfinal['cuisine'])

X_test = testfinal[features].values

In [99]: testfinal.dtypes
```

```
Out[99]: id                int64
week                int64
city_code           int64
region_code         int64
center_type         int64
op_area            float64
category            int64
cuisine             int64
checkout_price      float64
dtype: object
```



```
Out[99]: id                int64
week                int64
city_code           int64
region_code         int64
center_type         int64
op_area            float64
category            int64
cuisine             int64
checkout_price      float64
base_price          float64
emailer_for_promotion int64
homepage_featured   int64
dtype: object

In [100]: pred = DT.predict(X_test)
pred[pred<0] = 0
submit = pd.DataFrame({
    'id': testfinal['id'],
    'num_orders': pred
})

Submit the predicted output values(Number of orders) to "submission.csv"

In [101]: submit.to_csv("submission.csv", index=False)

In [102]: submit.describe()
```

```
Out[102]:
```

	id	num_orders
count	3.257300e+04	32573.000000

Food demand forecaster - IBM

dataplatform.cloud.ibm.com/analytics/notebooks/v2/0dbe901f-cf68-4bb0-a089-d5f17ae6376b?projectId=f19d26dc-a2cc-4898-9eca-d90167184e8c&context=cpdaas

IBM Watson Studio

Search in your workspaces

Buy

NISSHANTH JAYAMURALI...

Dallas

N

Projects / DemandEst-AI Powered Food De... / Food demand forecaster

File Edit View Insert Cell Kernel Help

Not Connected error Not Trusted Python 3.9

homepage_featured
dtype: object

int64

In [100]:

```
pred = DT.predict(X_test)
pred[pred<0] = 0
submit = pd.DataFrame({
    'id': testfinal['id'],
    'num_orders': pred
})
```

Submit the predicted output values(Number of orders) to "submission.csv"

In [101]:

```
submit.to_csv("submission.csv", index=False)
```

In [102]:

```
submit.describe()
```

Out[102]:

	id	num_orders
count	3.257300e+04	32573.000000
mean	1.248476e+06	262.904686
std	1.441580e+05	367.738525
min	1.000085e+06	15.500000
25%	1.123969e+06	64.910747
50%	1.247296e+06	147.792453
75%	1.372971e+06	322.843750
max	1.499996e+06	6892.500000

Type here to search

1539
19-11-2022