

TRAIN THE ML MODEL ON IBM

Team id: PNT2022TMID15311

TRAIN AND TEST MODEL ALGORITHMS

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

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.

```
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: 131.12700245932322

In [94]: 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: 62.843557611229585

In [95]: 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: 67.1868189545414

In [96]: 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)))

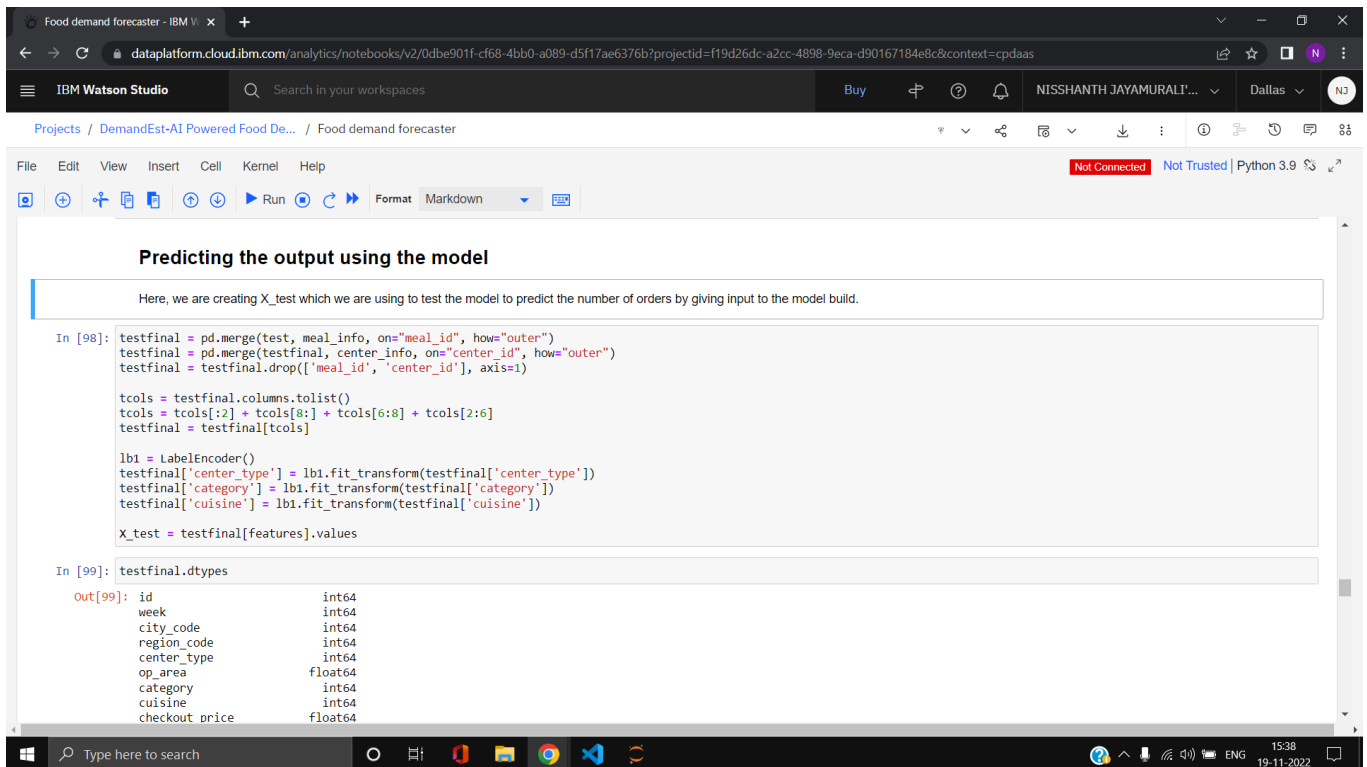
RMSLE: 97.77041683634049
```

SAVE THE MODEL

```
Save the Model

In [97]: import pickle
pickle.dump(DT, open('fdemand.pk1', 'wb'))
```

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.

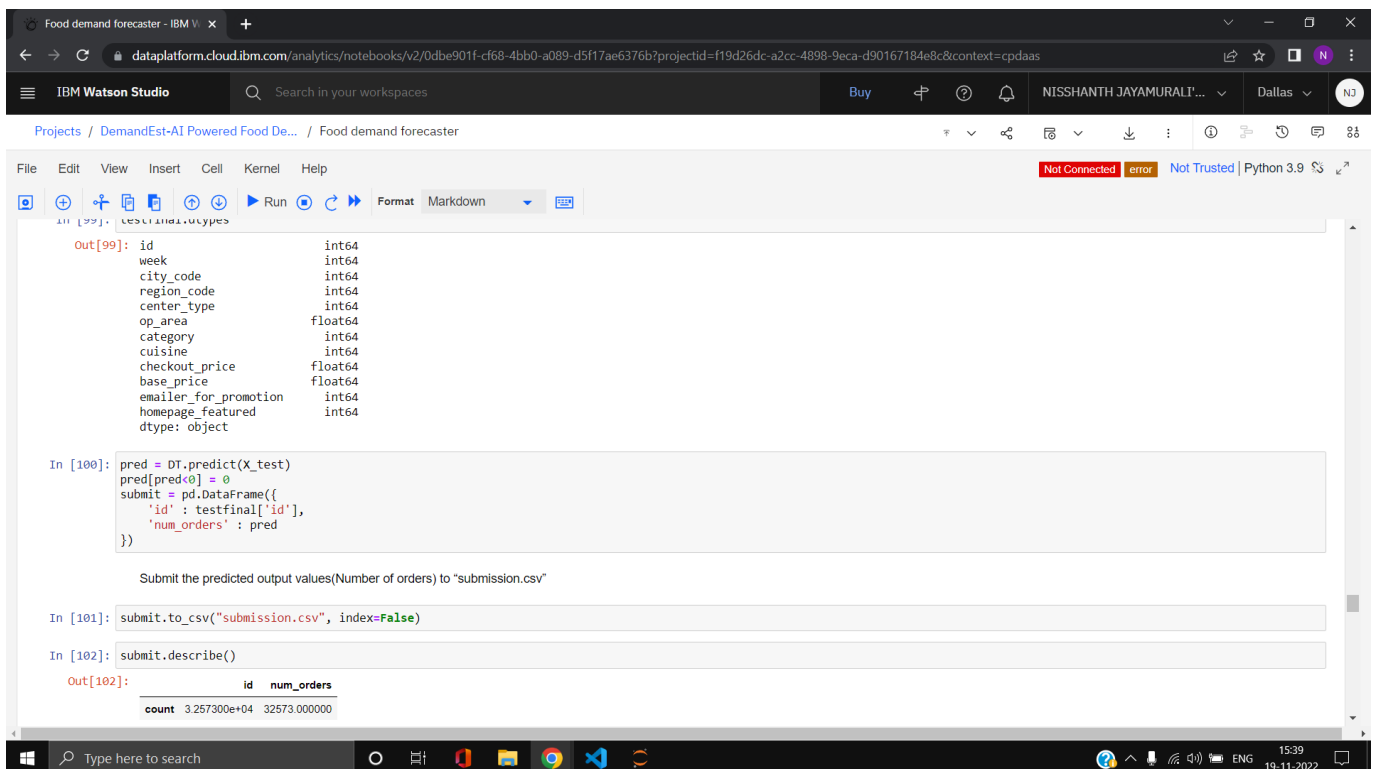
```
In [98]: testfinal = pd.merge(test, meal_info, on="meal_id", how="outer")
testfinal = pd.merge(testfinal, center_info, on="center_id", how="outer")
testfinal = testfinal.drop(['meal_id', 'center_id'], axis=1)

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



```
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()
```

	id	num_orders
Out[102]:		
count	3.257300e+04	32573.000000

