INVENTORY MANAGEMENT

GROUP-IV

a. *Gokul*
b. *Sudha*
c. *Anuradha*
d. *Abhishek*
e. *Anu Sanal*
f. *Ashwini*

01/05/2021

Mentors: Vinod and Bhanupriya

***Business Problem*:**

***Objective*:**

Poor inventory management leads to a loss in sales which in turn paints an inaccurate picture of lower demand for certain items, making future order predictions based on that past data inherently inaccurate. Instead, smart retailers use real-time data to move inventory where it's needed before it's too late. Additionally, they use predictive analytics to decide what to stock and where based on data about regional differences in preferences, weather, etc by using Python Technology.

Project Architecture / Project Flow

1. **Business Understanding**
   Define the business question
2. **Data Understanding**
   Get familiar with the data
3. **Data Preparation**
   Combine, transform, clean data
4. **Modeling**
   Apply algorithms, calibrate parameters
5. **Evaluation**
   Review and determine usability
6. **Deployment**
   Simple - Generate a report
   Complex - Implement a repeatable scoring process

Python Libraries Used in Project

Exploratory Data Analysis (EDA) and Feature Engineering

Data set details

There are 2 datasets

- Product details
- Product revenue

Product details:
Rows -1115
Columns - 3

```
Product dataset features :
 product_type      int64
cost_per_unit     int64
time_delivery     int64
```

Product revenue:
Rows -1017209
Columns – 8

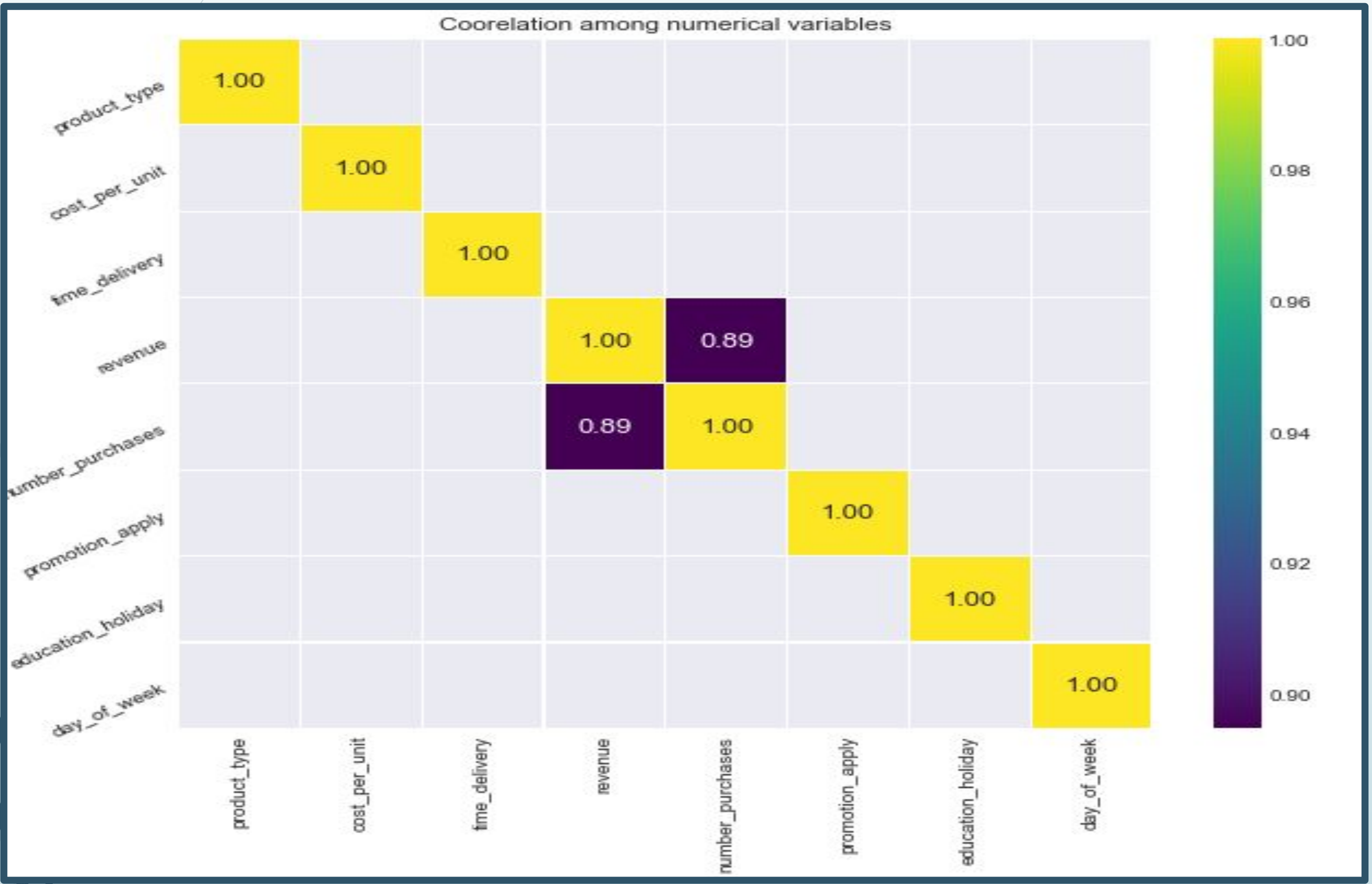
```
Revenue dataset features :
 product_type              int64
revenue                   int64
number_purchases          int64
store_status             object
promotion_apply           int64
generic_holiday          object
education_holiday         int64
day_of_week               int64
```

Exploratory Data Analysis (EDA)

- *Store_status and Generic holiday are character data type remaining other column are integer data type.*
- *No missing values in the dataset*
- *No. of duplicates – 156958*
- *Revenue is high when applying promotion*
- *For Outliers Handling we can use scaling techniques*

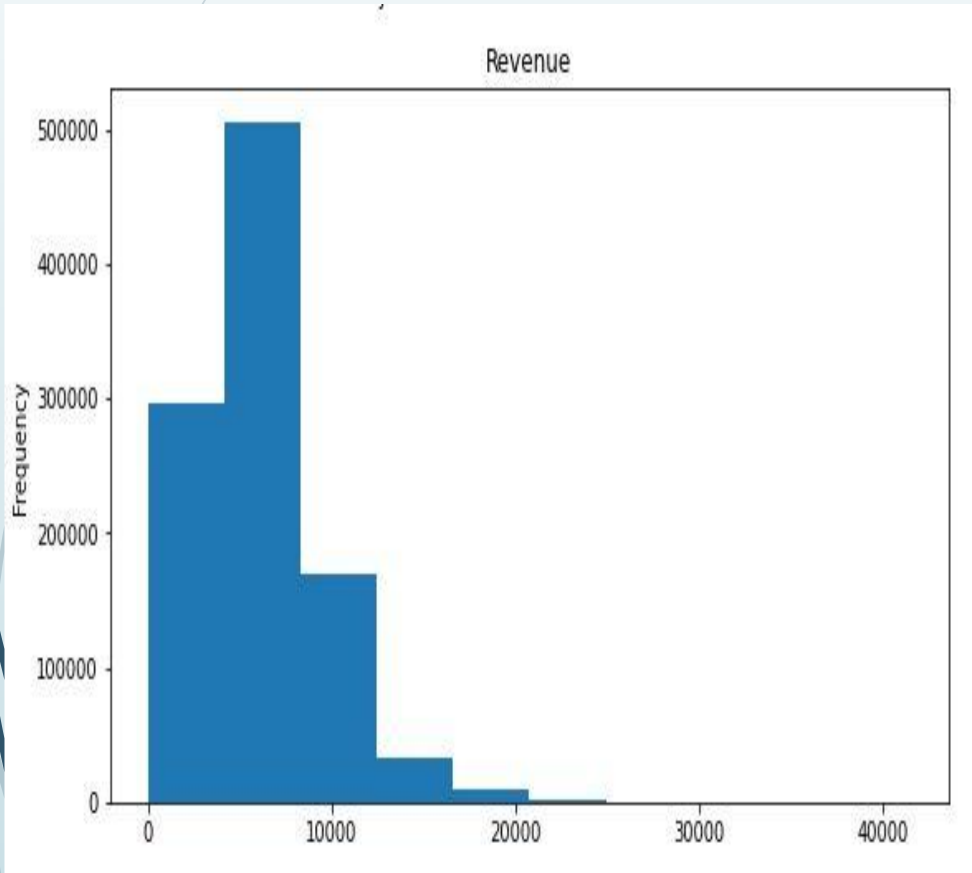
EDA

Correlation between no. of purchases and revenue high



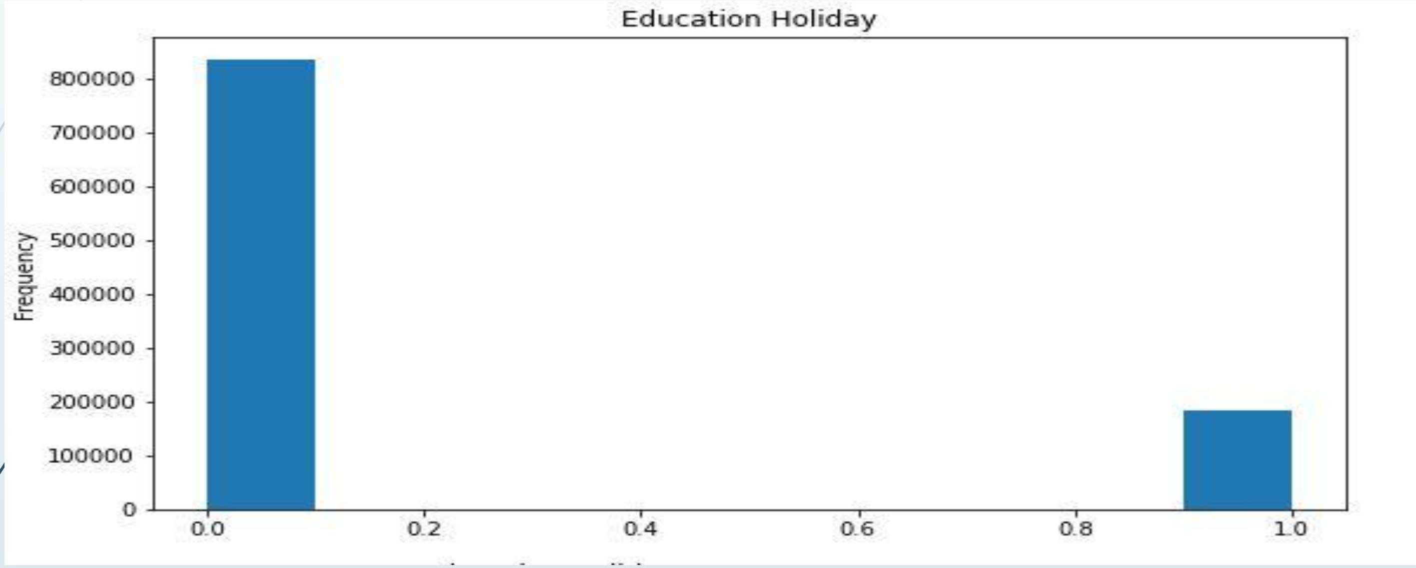
REVENUE HISTOGRAM PLOT

As per the below graph- Revenue is right skewed



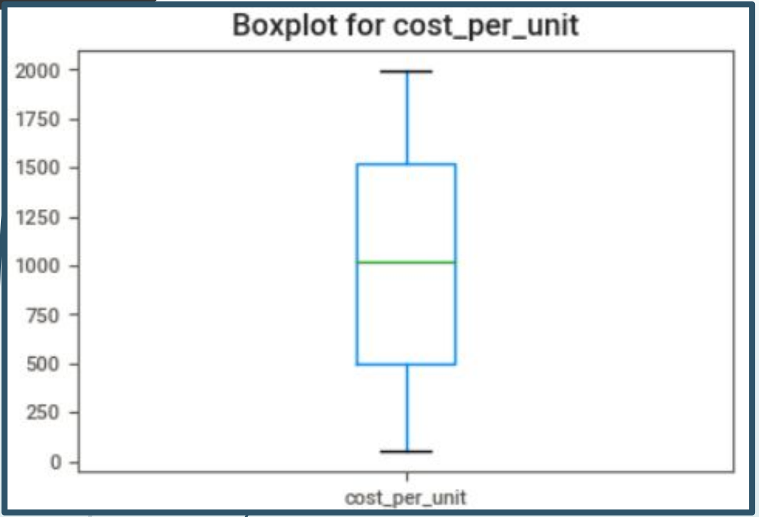

```
Numerical Column

 Revenue
count    1017209
mean        5773
std         3849
min            0
25%         3727
50%         5744
75%         7856
max        41551
Name: Revenue, dtype: int64
```

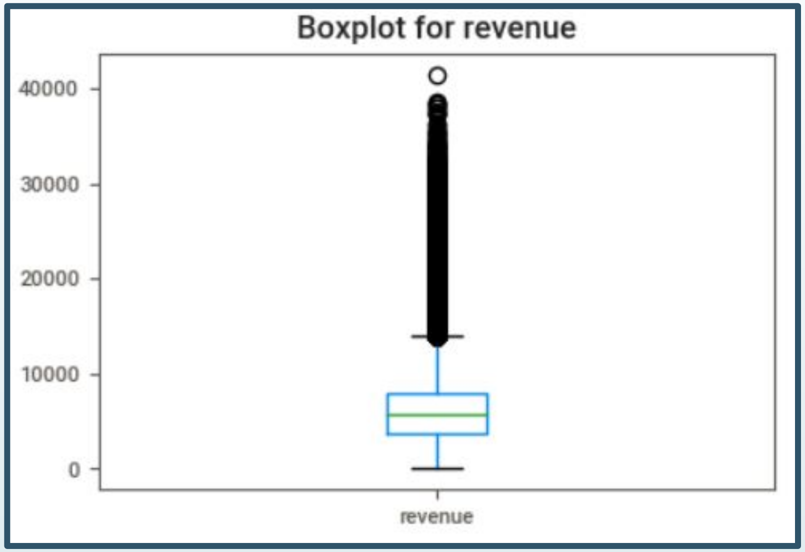
Histogram Plot for Education Holiday and counts for store status

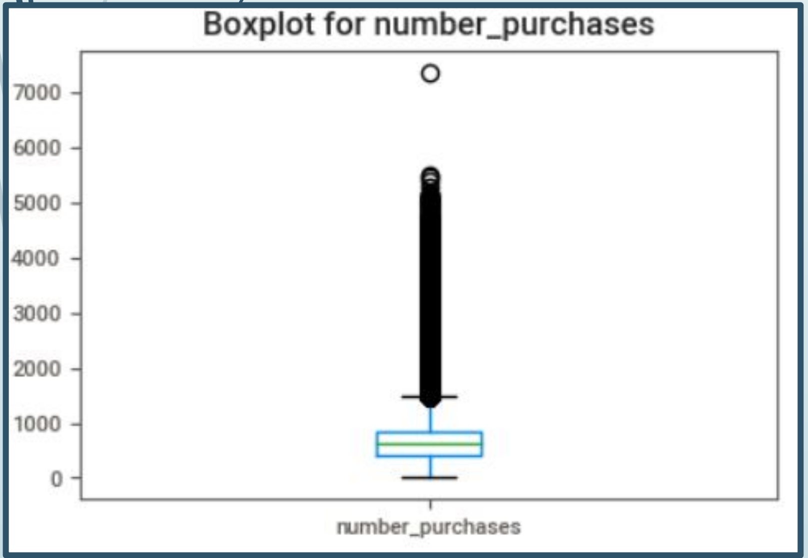


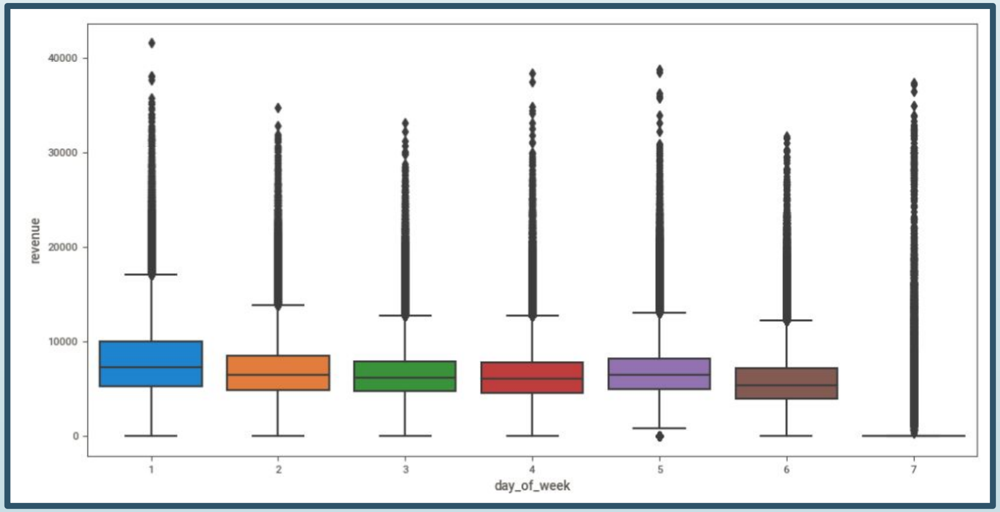
- *Merging the two datasets Product details and product revenue with respect to product type.*

	count	mean	std	min	25%	50%	75%	max
product_type	1017209.00	558.43	321.91	1.00	280.00	558.00	838.00	1115.00
cost_per_unit	1017209.00	1012.84	565.50	50.00	502.00	1023.00	1519.00	1999.00
time_delivery	1017209.00	9.54	2.86	5.00	7.00	10.00	12.00	14.00
revenue	1017209.00	5773.83	3849.95	0.00	3727.00	5744.00	7856.00	41551.00
number_purchases	1017209.00	633.14	464.41	0.00	405.00	609.00	837.00	7388.00
promotion_apply	1017209.00	0.38	0.49	0.00	0.00	0.00	1.00	1.00
education_holiday	1017209.00	0.18	0.38	0.00	0.00	0.00	0.00	1.00
day_of_week	1017209.00	4.00	2.00	1.00	2.00	4.00	6.00	7.00

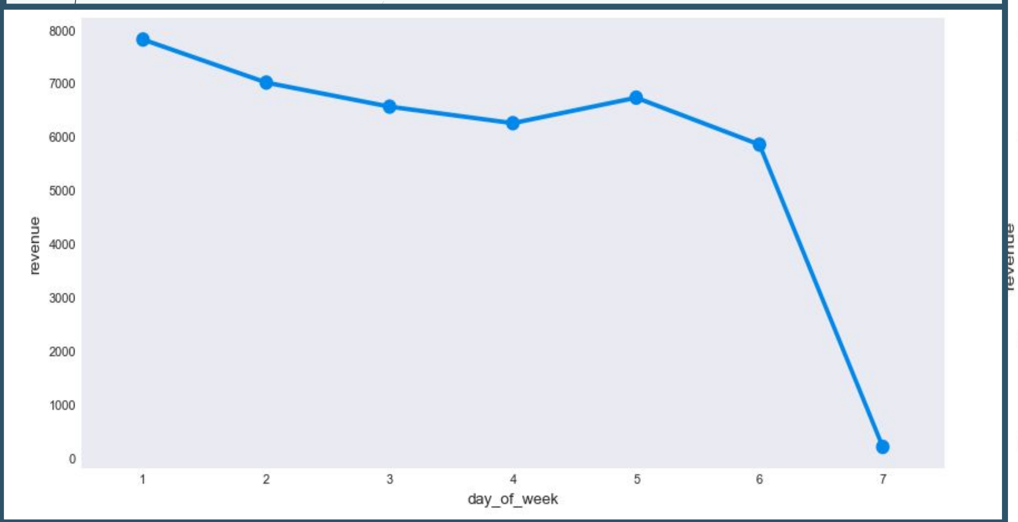

Boxplot for cost_per_unit
2000
1750
1500
1250
1000
750
500
250
0
cost_per_unit

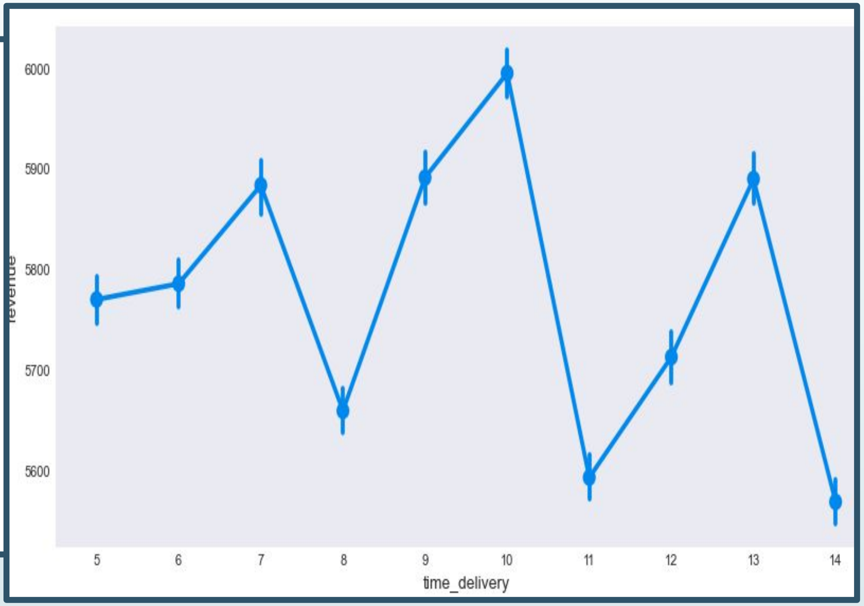

Boxplot for revenue
40000
30000
20000
10000
0
revenue

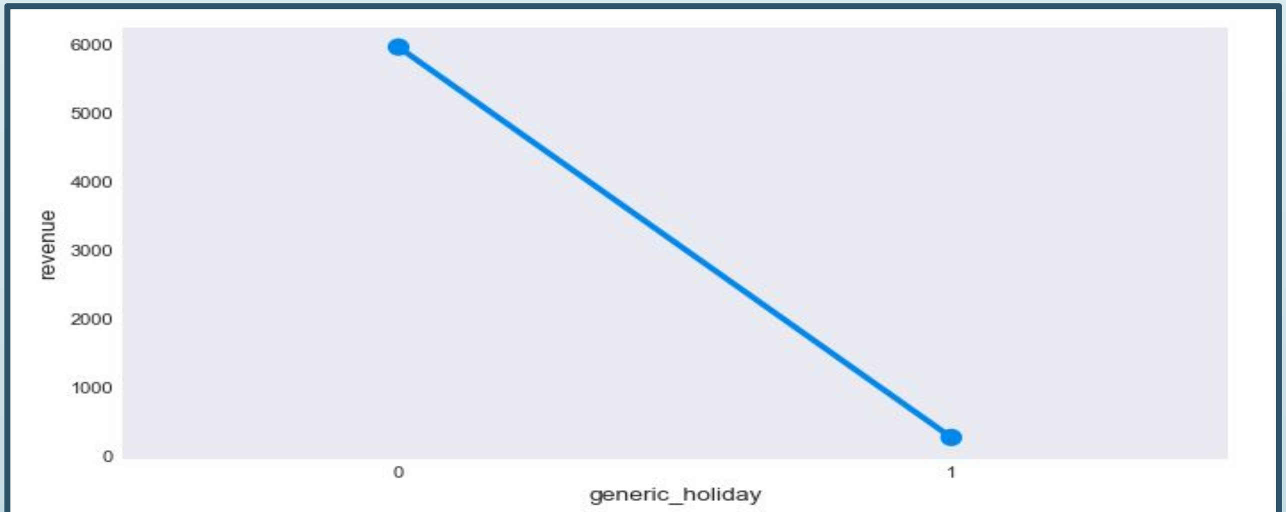

Boxplot for number_purchases
7000
6000
5000
4000
3000
2000
1000
0
number_purchases


revenue
40000
30000
20000
10000
0
1
2
3
4
5
6
7
day_of_week

POINT PLOT AGAINST REVENUE







Feature Engineering

Feature engineering has two goals primarily:

- Preparing the proper input dataset, compatible with the machine learning algorithm requirements
- Improving the performance of machine learning models

Following categorical columns converted into numerical
- Store_status

Feature Engineer target Variable for feeding dataset to ML algorithms.

TARGET VARIABLE:

Calculated target variable column no. of units using the below formula

No.of units = revenue / cost per unit

- Adding 10% buffer to no. of units to avoid shortage/ overload of inventory

- Normalised the dataset independent variables using Robustscaler Technique

Feature Selection by RFE and Decision Tree techniques

Decision Tree Feature Importance on whole dataset

	Columns	Feature_IMP
0	product_type	0.00015
1	cost_per_unit	0.79522
2	time_delivery	0.00006
3	revenue	0.20440
4	number_purchases	0.00015
5	store_status	0.00000
6	promotion_apply	0.00000
7	generic_holiday	0.00000
8	education_holiday	0.00000
9	day_of_week	0.00002

RFE Feature Ranking

	Columns	Feature_IMP
0	product_type	3.108101e+02
1	cost_per_unit	3.807241e+05
2	time_delivery	2.339764e+03
3	revenue	8.114169e+04
4	number_purchases	5.553236e+04
5	store_status	6.869101e+03
6	promotion_apply	9.530650e+03
7	generic_holiday	5.193270e+03
8	education_holiday	2.217300e+00
9	day_of_week	2.012102e+03
10	number_of_units	2.140451e+19

Feature Selection

- **Drop Generic_Holiday and Education_Holiday col**
- **Drop Number_of_Purchases col to remove collinearity problem with revenue column**

Model Building

Model results

Model –
RANDOM FOREST

Data set details

5% Sample of the whole data set

Data Partition details

TRAINING SET – 80%

TESTING SET – 20%

Algorithms

Partitioned the dataset with 80% and 20% for training and test. Scaled the features by Robust Scaler .Installed sklearn.ensemble package to unpack RandomForestRegressor function. Created model and calculated RMS.

Algorithm details and configuration

```
from sklearn.ensemble import
RandomForestRegressor

regressor_RF =
RandomForestRegressor(n_estimators = 20,
random_state = 0)
regressor_RF.fit(X_train, Y_train)
```

Root Mean Squared Error – 1.1068

```
Mean Absolute Error     : 0.150696224382912
Mean Squared Error      : 1.225070148266235
Root Mean Squared Error : 1.106828870361735
```

Model results

EXCELR
Raising Excellence

Model – DECISION TREE

Data set details

5% Sample of the whole data set

Data Partition details

TRAINING SET – 80%

TESTING SET – 20%

Algorithms

Partitioned the dataset with 80% and 20% for training and test.Scaled the features by Robust Scaler Installed sklearn.tree package to unpack DecisionTreeRegressor function. Created model and calculated RMS.

Algorithm details and configuration

```
from sklearn.tree import DecisionTreeRegressor

regressor_DT = DecisionTreeRegressor()
regressor_DT.fit(X_train, Y_train)
```

Root Mean Squared Error – 1.081

```
Mean Absolute Error      : 0.2316785132605774
Mean Squared Error       : 1.169636694110116
Root Mean Squared Error  : 1.0814974313932122
```

Model results

EXCELR
Raising Excellence

Model – LINEAR REGRESSION

Data set details

5% Sample of the whole data set

Data Partition details

TRAINING SET – 80%

TESTING SET – 20%

Algorithms

Partitioned the dataset with 80% and 20% for training and test.Scaled the features by Robust Scaler. Installed sklearn.linear package to unpack LinearRegressor function. Created model and calculated RMS.

Algorithm details and configuration

```
from sklearn.linear_model import LinearRegression

regressor_LR = LinearRegression()
regressor_LR.fit(X_train, Y_train)
```

Root Mean Squared Error – 17.313

V

```
Mean Absolute Error      : 9.343981496902034
Mean Squared Error       : 299.74668048123885
Root Mean Squared Error  : 17.31319382671028
```

Model results

EXCELR
Raising Excellence

Model – NN Regressor

Data set details

5% Sample of the whole data set

Data Partition details

TRAINING SET – 80%

TESTING SET – 20%

Algorithms

Partitioned the dataset with 80% and 20% for training and test.Scaled the features by Robust Scaler. Installed keras package to create NN model. Created model and calculated RMS.

Algorithm details and configuration

```
from keras.models import Sequential
from keras.layers import Dense, Activation, Flatten

NN_model.fit(X_train,Y_train, epochs=10, batch_size=50, validation_split = 0.2)
```

Root Mean Squared Error – 25.17

V

```
Mean Absolute Error      : 13.780497547642756
Mean Squared Error       : 633.7768590176254
Root Mean Squared Error : 25.17492520381392
```

Model results

EXCELR
Raising Excellence

Model –
RANDOM FOREST

Data set details

whole data set

Data Partition details

TRAINING SET – 75%

TESTING SET – 25%

Algorithms

Partitioned the dataset with 75% and 25% for training and test. Scaled the features with Robust Scaler .Installed sklearn.ensemble package to unpack RandomForestRegressor function. Created model and calculated RMS.

Algorithm details and configuration

```
from sklearn.ensemble import
RandomForestRegressor

regressor_RF =
RandomForestRegressor(n_estimators = 20,
random_state = 0)
regressor_RF.fit(X_train, Y_train)
```

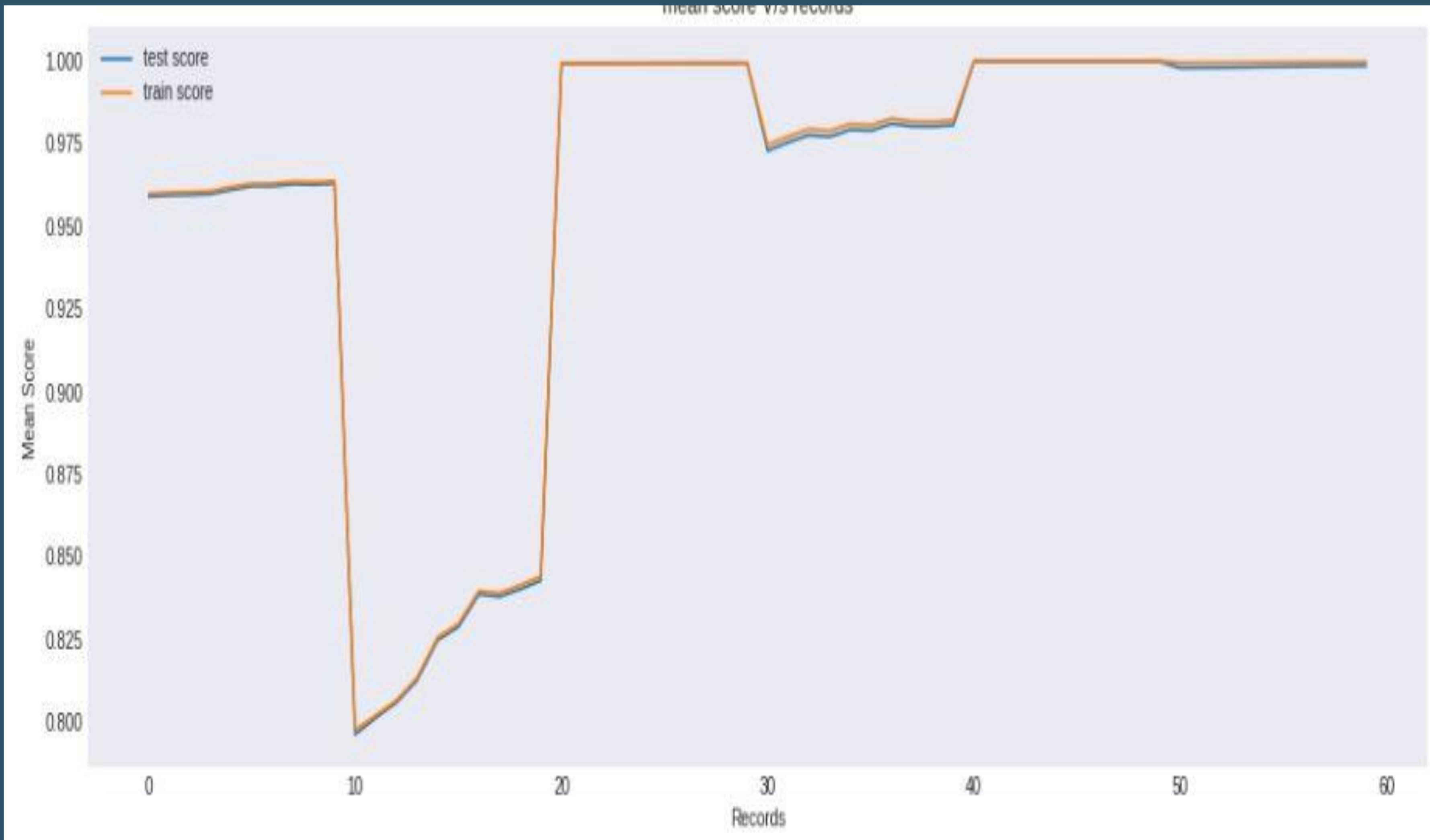
Root Mean Squared Error – 0.2415

```
Mean Absolute Error     : 0.01925158937258893
Mean Squared Error      : 0.05835326944124611
Root Mean Squared Error : 0.24156421390853014
```

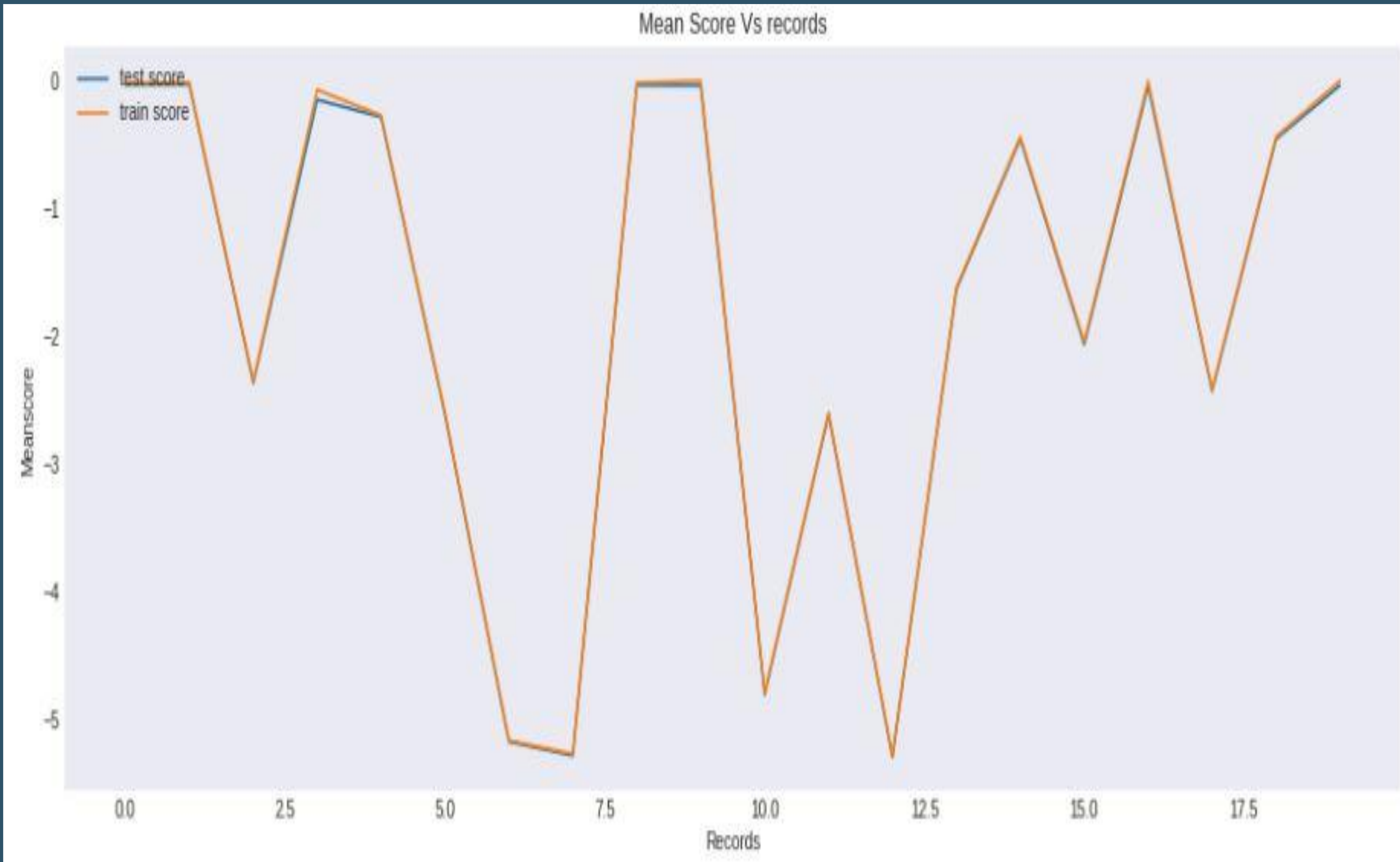
RandomForest is best model as it is giving less RMSE value

Hyperparameter tuning and Cross Validation of Radom Forest Model

Grid Search CV



RandomisedSearch CV



RESULTS:

- **print(model_cv_rf.best_params_) #Grid Search CV**
- **{'bootstrap': True, 'max_depth': None, 'max_features': 'auto', 'n_estimators': 13}**
- **print(model_RSCV_rf.best_params_)**
- **#RandomisedSearch CV**
- **{'n_estimators': 12, 'max_features': 'auto', 'max_depth': None, 'bootstrap': True}**
- **We Select**
- **regressor_RF = RandomForestRegressor(n_estimators= 12,max_features = 'auto', max_depth = None, bootstrap=True)**
- **regressor_RF.fit(X_train, Y_train)**

Pycaret Package Results

exp_reg101 = setup(data = sample_data, target = 'per_added', session_id=105,remove_outliers = True, outliers_threshold = 0.05, normalize = True, normalize_method = 'robust', transformation = True, html = False)

best_specific = compare_models(include = ['lr','lasso','ridge','knn','dt','rf','et', 'lightgbm'])

Pycaret Package Results

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
rf	Random Forest Regressor	0.1196	0.7561	0.7636	0.9985	0.0129	0.0049	10.844
dt	Decision Tree Regressor	0.2343	1.3262	1.1002	0.9973	0.0175	0.0122	0.192
knn	K Neighbors Regressor	2.7963	99.9123	9.9833	0.7939	0.1869	0.1393	0.473
lr	Linear Regression	9.2463	295.3055	17.1564	0.3963	0.9778	1.0393	0.295
ridge	Ridge Regression	9.2460	295.3054	17.1564	0.3963	0.9778	1.0392	0.030
lasso	Lasso Regression	8.6672	299.9098	17.2855	0.3874	0.9385	0.8844	0.031

Final Random Forest Model

from sklearn.ensemble import RandomForestRegressor

regressor_RF = RandomForestRegressor(n_estimators= 12,max_features = 'auto', max_depth = None, bootstrap=True)

regressor_RF.fit(X_train, Y_train)

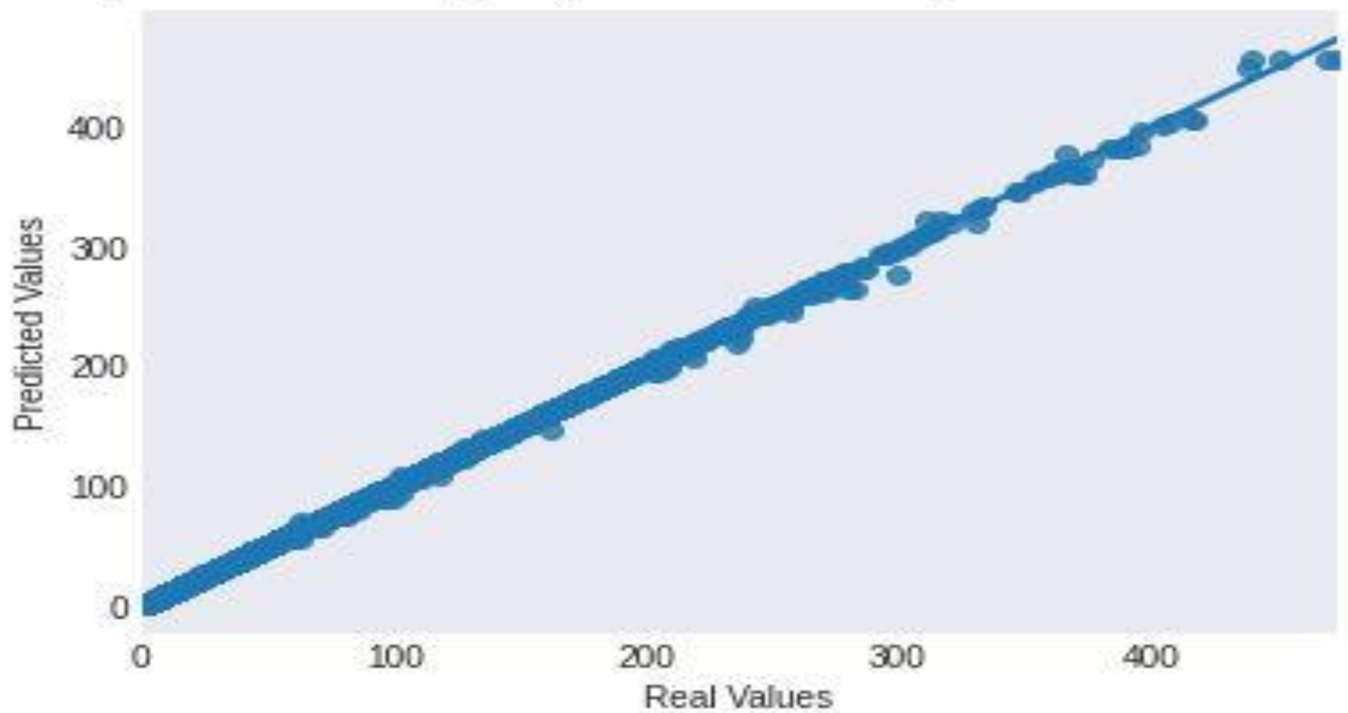
y_pred_RF = regressor_RF.predict(X_test)

```
Mean Absolute Error      : 0.01936792359746316
Mean Squared Error       : 0.03058347237898626
Root Mean Squared Error : 0.1748813094043679
```

Final Random Forest Model

TEST DATASET PREDICTED VS ACTUAL VALUES PLOT

<matplotlib.axes._subplots.AxesSubplot at 0x7effbbbef950>



Final Random Forest Model

- **Pickled the finalized Model**
- **Checked the score on test dataset again**

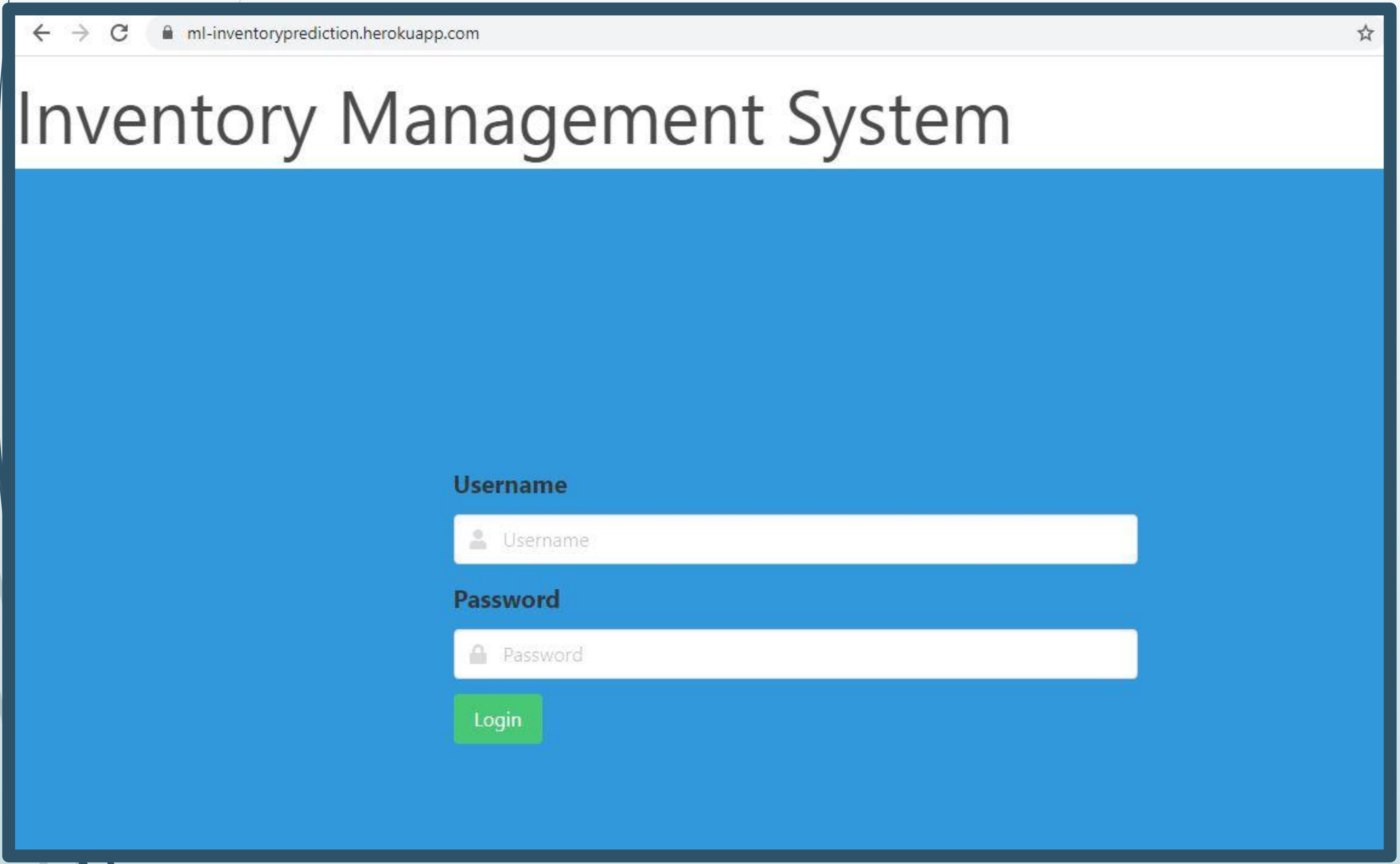
```
# load the model from disk
loaded_model = pickle.load(open(filename, 'rb'))
result = loaded_model.score(X_test, Y_test)
print(result)

0.9999372337963354
```

Model Deployment In Local and Cloud

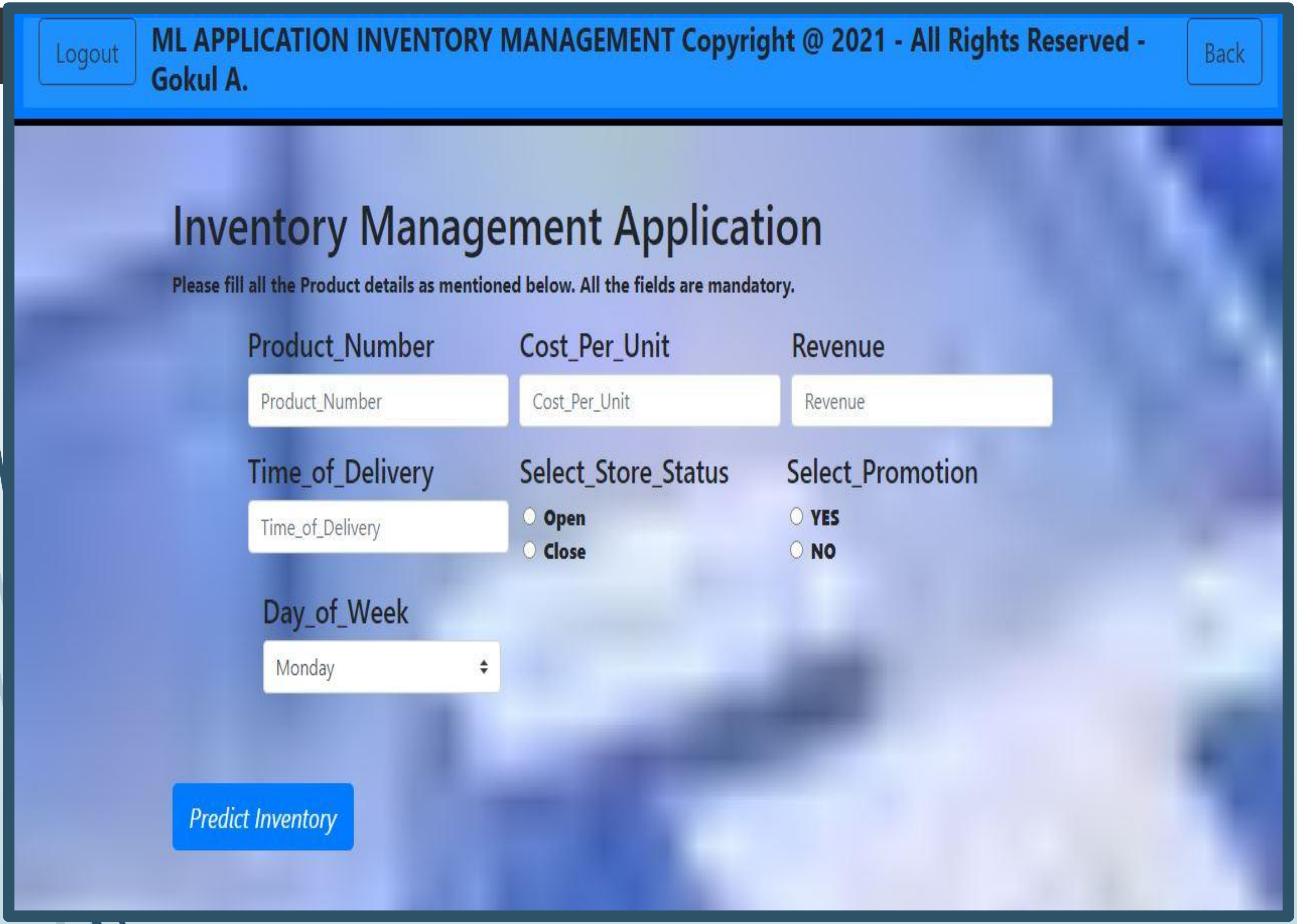
Heroku Deployment link

Using Flask Framework for Deployment

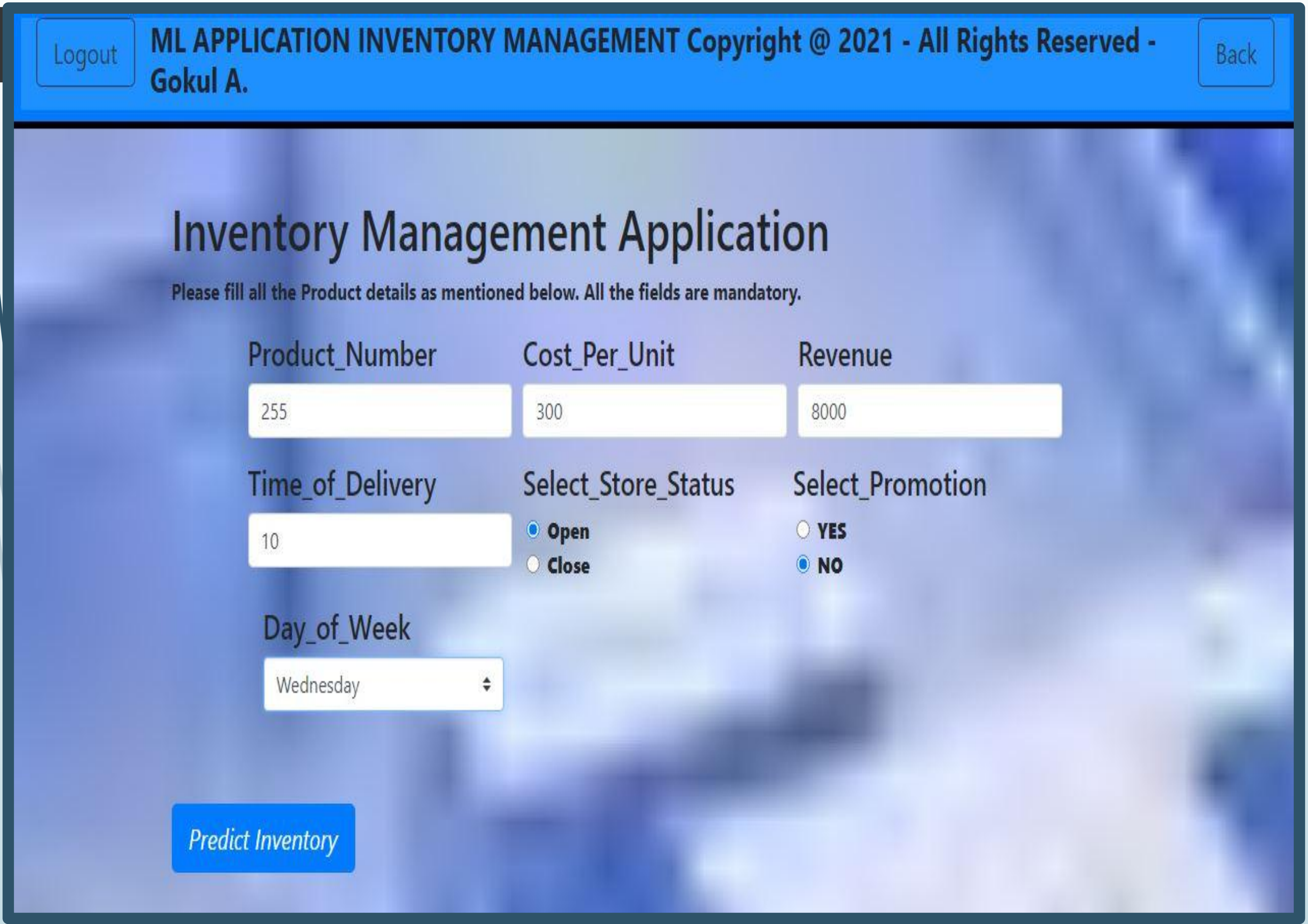
A. Login page



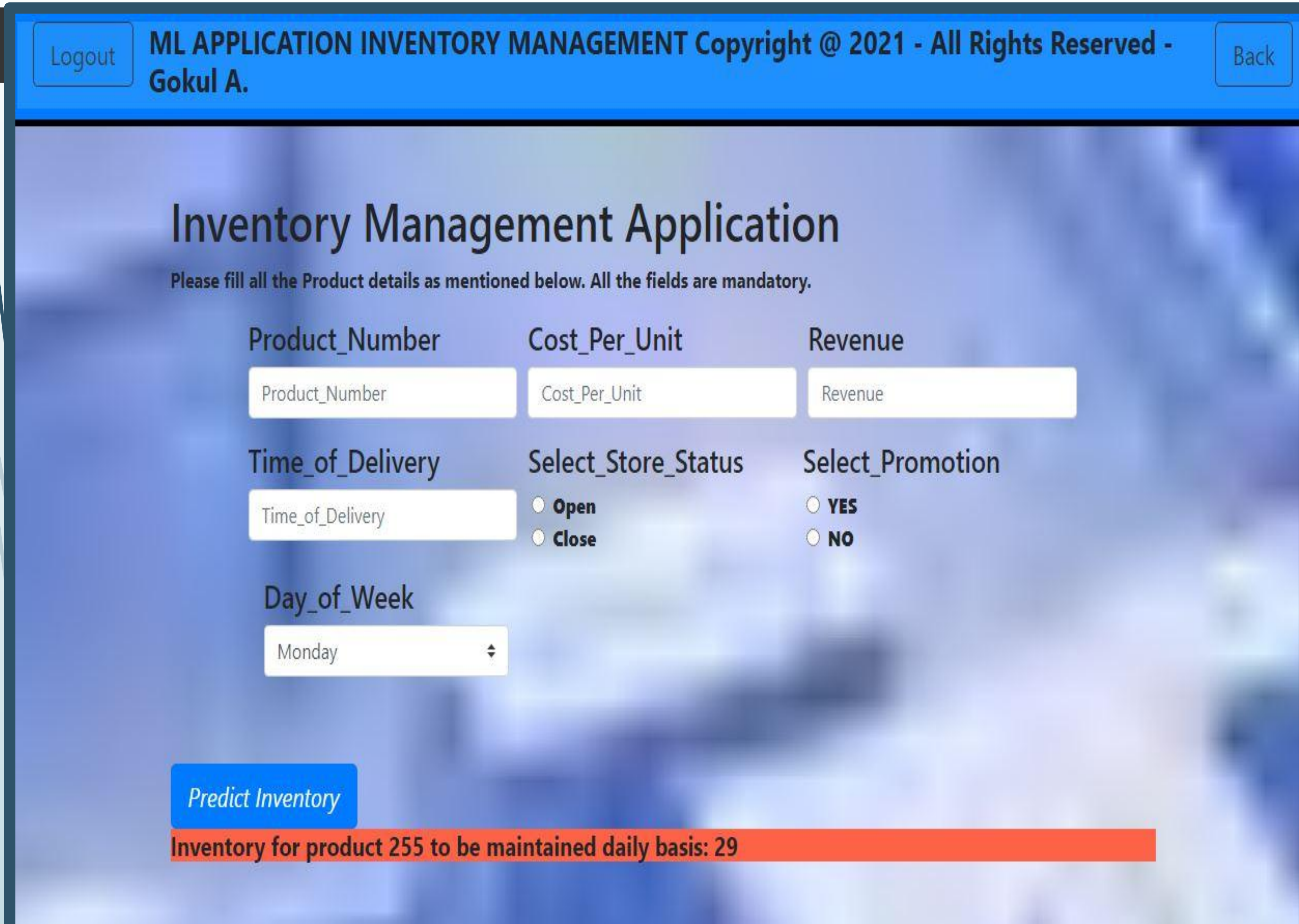
B. Application Page Before Field Values



C. Application Page After Field Values



D. Application Page After Clicking Predict Button



EXCELR
Raising Excellence

Thank you