Mega store

Team ID: SC_14

Team members:

	Name	University ID
1	محمد عبدالرؤف أحمد عبد الشافي	20191700557
2	فاطمة محمود عبد الحكم موسى بدر	20201700571
3	مصطفى محمد بيومي محمد	20201700837
4	مصطفى هانى محمد محروس	20201700843
5	عمر محمد عبدالرؤوف إبراهيم	20201700545
6	رحمة ايمن عطية عوض	20201700257

Milestone 1

Report

First: Splitting data:

- we have divided the data into two parts **: X(input features)** and **Y(target)** then we divided each of them into **train(80%)** and **test(20%),**and we have used **shuffle parameter** to shuffle the data randomly before splitting to avoid any order biases. Finally**, the random_state parameter** is set to 42, which ensures that the split will be the same every time the code is run.
- After split the entire data to train and test we call pre_processing(X_train) which explained below.

Second: preprocessing data:

- We have implemented a preprocessing.py which contains three functions.
  - pre_processing(X)
    - This function takes the data frame of features and makes some changes to it and then returns it again.
    - the changes are represented in:
      1- split columns of date {'Order Date' , 'Ship Date'} each to three columns day , month , year.
      2- Split the column => 'Category tree' which data type is Dictionary to number of columns same as the number of unique Keys is all samples.
    - The output data frame contains 8 new columns, and we drop the old 3 columns so its net size increase by 5 columns.
  - numerical_Categorical(X:pd.dataframe())
    - This function takes a data frame of features and returns the numerical and categorical columns as two data frames.
  - date_split(X:pd.dataframe , cols)
    - This function takes two parameters:
      - X: data frame which contains a date column.
      - cols: List of string holds the names of columns which contain date.
    - split columns of date each to three columns day, month, and year.

- ## Encoding:
    - **After some experiments we decide to use The Ordinal Encoder from the categorical_encoders library with two arguments:**
        - handle_unknown='value'   and   handle_missing='value'
        - We used handle_unknown argument to handle categories in the test set that has not seen in the training set.
        - We used handle_missing argument to handle missing values in the data

- # Feature selection:

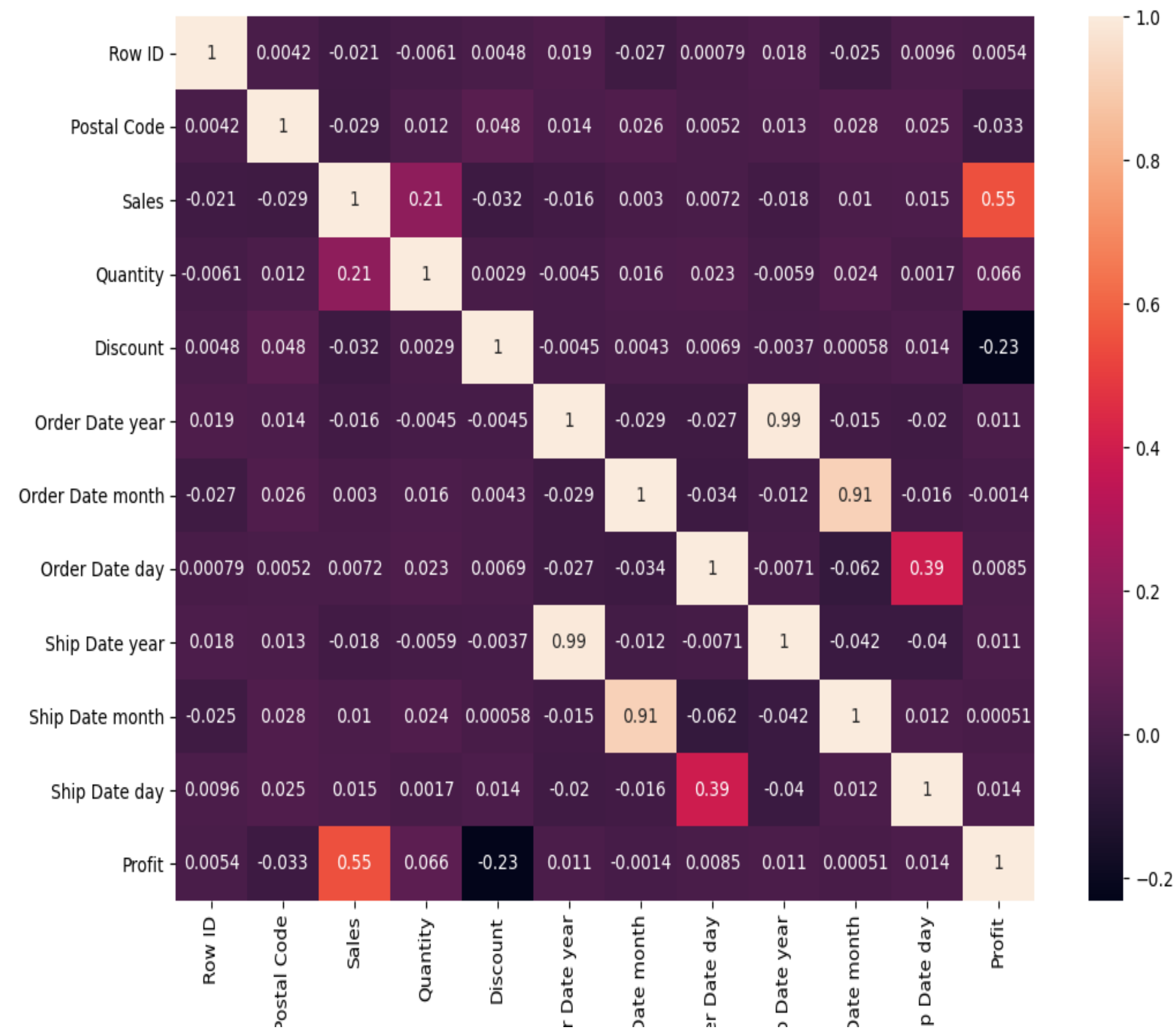
After separating data into categorical and numerical we made feature selection on each one

1- **SelectKBest** method from the scikit-learn library to select the top k features based on their F-scores, The F-score measures the linear dependence between the target variable and each feature in the categorical data to find out the most important of columns, then we have taken 6 of them.

~ Selected categorical features: main Category, subcategory, state, region, ship Mode, product name.

2- **Correlation** method for numerical data by selecting the top features that have a correlation coefficient greater than 0.2 with the 'Profit' variable We used correlation to find out the variables that are highly related to the target variable and can help in building a better predictive model**.**

~ Selected numerical features: sales, discount



Row ID
Postal Code
Sales
Quantity
Discount
Order Date year
Order Date month
Order Date day
Ship Date year
Ship Date month
Ship Date day
Profit
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

Third: Regression models:

Model	Train MSE	Test MSE	Model score
Polynomial model Degree(3)	2388.3440	2317.6800	0.9366
Random forest regressor Max_depth = 12	2067.4593	3185.283	0.9129
elasticNet model	33731.2383	19050.1557	0.4791
Ridge model	31664.5149	18683.0855	0.4892
Lasso model	31693.5716	18507.0796	0.4939
Linear-multivariable model	31664.4741	18690.8967	0.4889

```
mean square error of elasticNet train data set: 33731.238398328256
mean square error of elasticNet test data set: 19050.155687498733
elasticNet score :  0.47912795442623324

mean square error of ridge train data set: 31664.514973292124
mean square error of ridge test data set: 18683.085507303163
ridge model score :  0.48916443910194984

mean square error of lasso train data set: 31693.571627158366
mean square error of lasso test data set: 18507.07959326607
lasso model score :  0.49397681764527823

mean square error of Linear-multivariable train data set: 31664.474112899126
mean square error of Linear-multivariable test data set: 18690.896749667652
Linear-multivariable model score  :  0.4889508630107301

mean sqr error of random-forest-regressor train data set: 2067.4593162503174
mean square error of random forest regressor test data set: 3185.283172843259
random forest regressor model score  :  0.9129075379127042

mean square error of Polynomial train data set: 2388.3439852475585
mean square error of Polynomial test data set: 2317.68000367696
Polynomial model Score :  0.9366296662187993
```

Train and test size for each:

Lasso model:

Train size: 6396 samples and 8 features

Test size: 1599 samples and 8 features

Ridge model:

Train size: 6396 samples and 8 features

Test size: 1599 samples and 8 features

elasticNet model:

Train size: 6396 samples and 8 features

Test size: 1599 samples and 8 features

Linear multivariable model:

Train size: 6396 samples and 8 features

Test size: 1599 samples and 8 features

Random forest model:

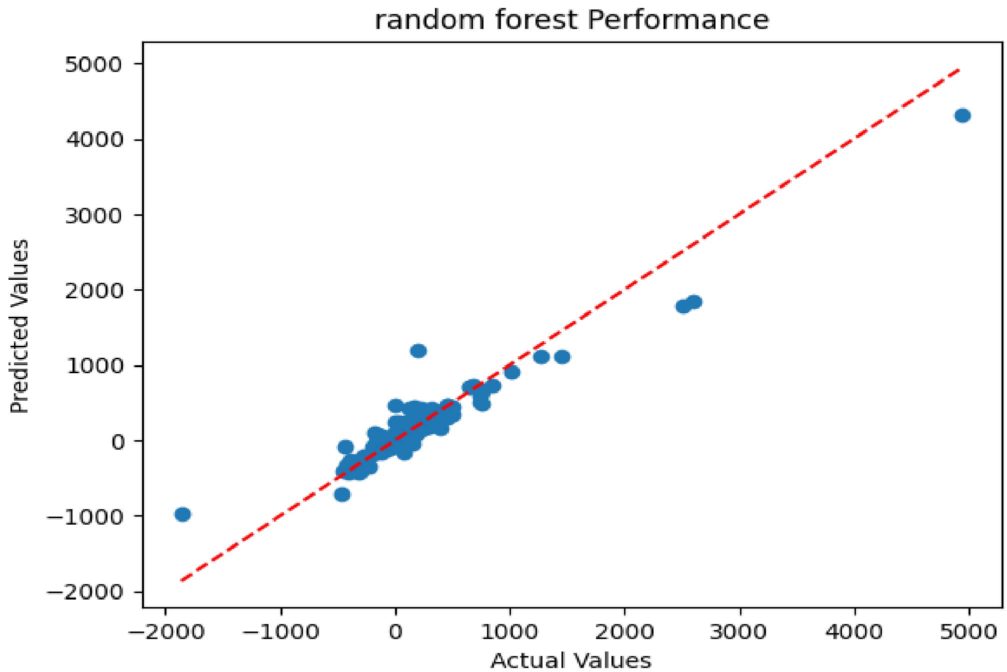
Train size: 6396 samples and 8 features

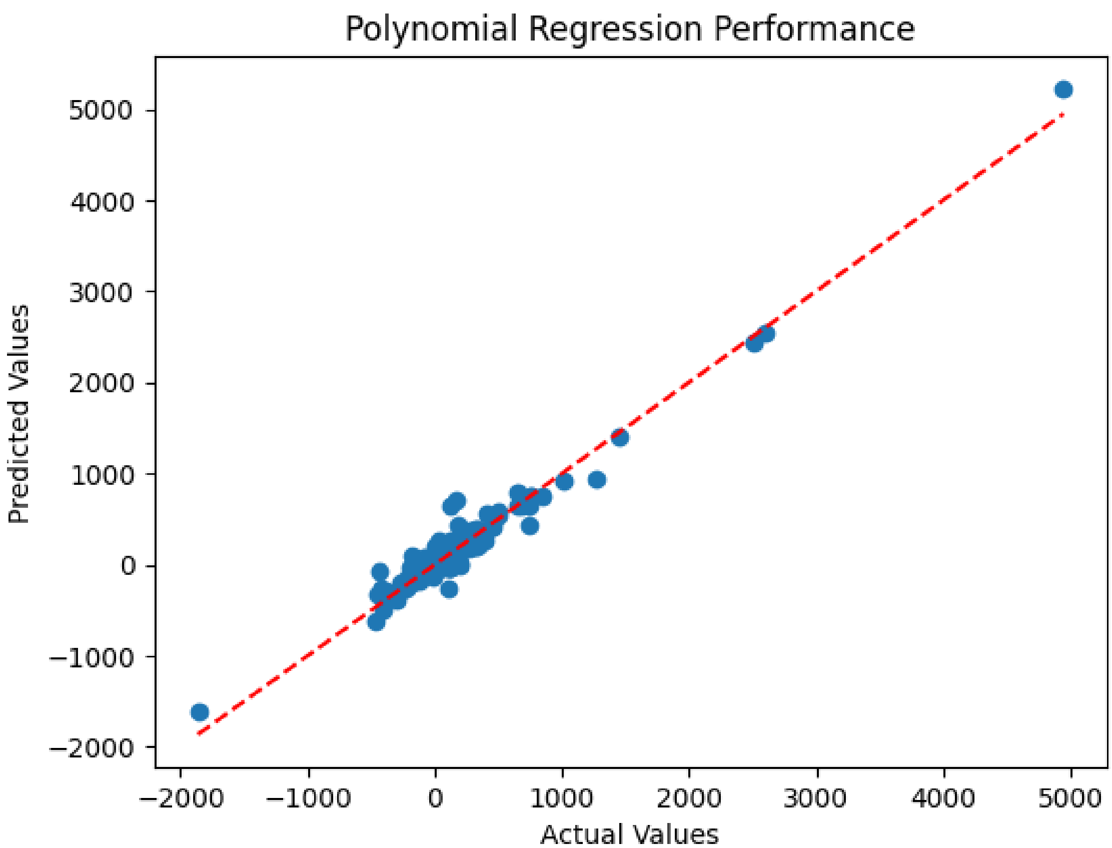
Test size: 1599 samples and 8 features

Polynomial model:

Train size: 6396 samples and 165 features

Test size: 1599 samples and 165 features


random forest Performance
Predicted Values
Actual Values
5000
4000
3000
2000
1000
0
−1000
−2000
−2000
−1000
0
1000
2000
3000
4000
5000


Polynomial Regression Performance
Predicted Values
Actual Values
5000
4000
3000
2000
1000
0
−1000
−2000
−2000
−1000
0
1000
2000
3000
4000
5000

Conclusion

- At the end we find that the polynomial model with degree = 3 is the best model to fit in our task that it has the minimum mean square error for training and testing.
- The random forest regressor is close to the polynomial model.
- The linear, Lasso, ridge and elasticNet models are very bad and their accuracy is less than 50%