Retail Capstone Project

DESCRIPTION

Problem Statement

- Demand Forecast is one of the key tasks in Supply Chain and Retail Domain in general. It is key in effective operation and optimization of retail supply chain. Effectively solving this problem requires knowledge about a wide range of tricks in Data Sciences and good understanding of ensemble techniques.
- You are required to predict sales for each Store-Day level for one month. All the features will be provided and actual sales that happened during that month will also be provided for model evaluation.

About Dataset

Training Data Description: Historic sales at Store-Day level for about two years for a retail giant, for more than 1000 stores. Also, other sale influencers like, whether on a particular day the store was fully open or closed for renovation, holiday and special event details, are also provided.

1. Project Task: Week 1

Exploratory Data Analysis (EDA) and Linear Regression:

1. **Transform the variables by using data manipulation techniques like, One-Hot Encoding.**

```
train_data.StateHoliday.unique() ## checking unique values
```

```
array(['0', 'a', 'b', 'c', 0], dtype=object)
```

```
train_data.loc[train_data.StateHoliday==0,'StateHoliday'] = '0'
```

- use One-Hot Encoding to convert this column

```
labelencoder= LabelEncoder()
train_data.StateHoliday = labelencoder.fit_transform(train_data['StateHoliday'])
```

```
train_data.StateHoliday.unique()
```

```
array([0, 1, 2, 3])
```

2. Perform an EDA (Exploratory Data Analysis) to see the impact of variables over Sales.

- Check the head and info of the dataset, after this checking null values in the dataset.

```
train_data.head()
```

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday
0	1	2	2015-06-30	5735	568	1	1	0	0
1	2	2	2015-06-30	9863	877	1	1	0	0
2	3	2	2015-06-30	13261	1072	1	1	0	1
3	4	2	2015-06-30	13106	1488	1	1	0	0
4	5	2	2015-06-30	6635	645	1	1	0	0

```
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 982644 entries, 0 to 982643
Data columns (total 9 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   Store          982644 non-null  int64
 1   DayOfWeek      982644 non-null  int64
 2   Date           982644 non-null  datetime64[ns]
 3   Sales          982644 non-null  int64
 4   Customers      982644 non-null  int64
 5   Open           982644 non-null  int64
 6   Promo          982644 non-null  int64
 7   StateHoliday   982644 non-null  object
 8   SchoolHoliday  982644 non-null  int64
dtypes: datetime64[ns](1), int64(7), object(1)
memory usage: 67.5+ MB
```

- StateHoliday column is object type

```
train_data.isna().sum() # checking null values
```

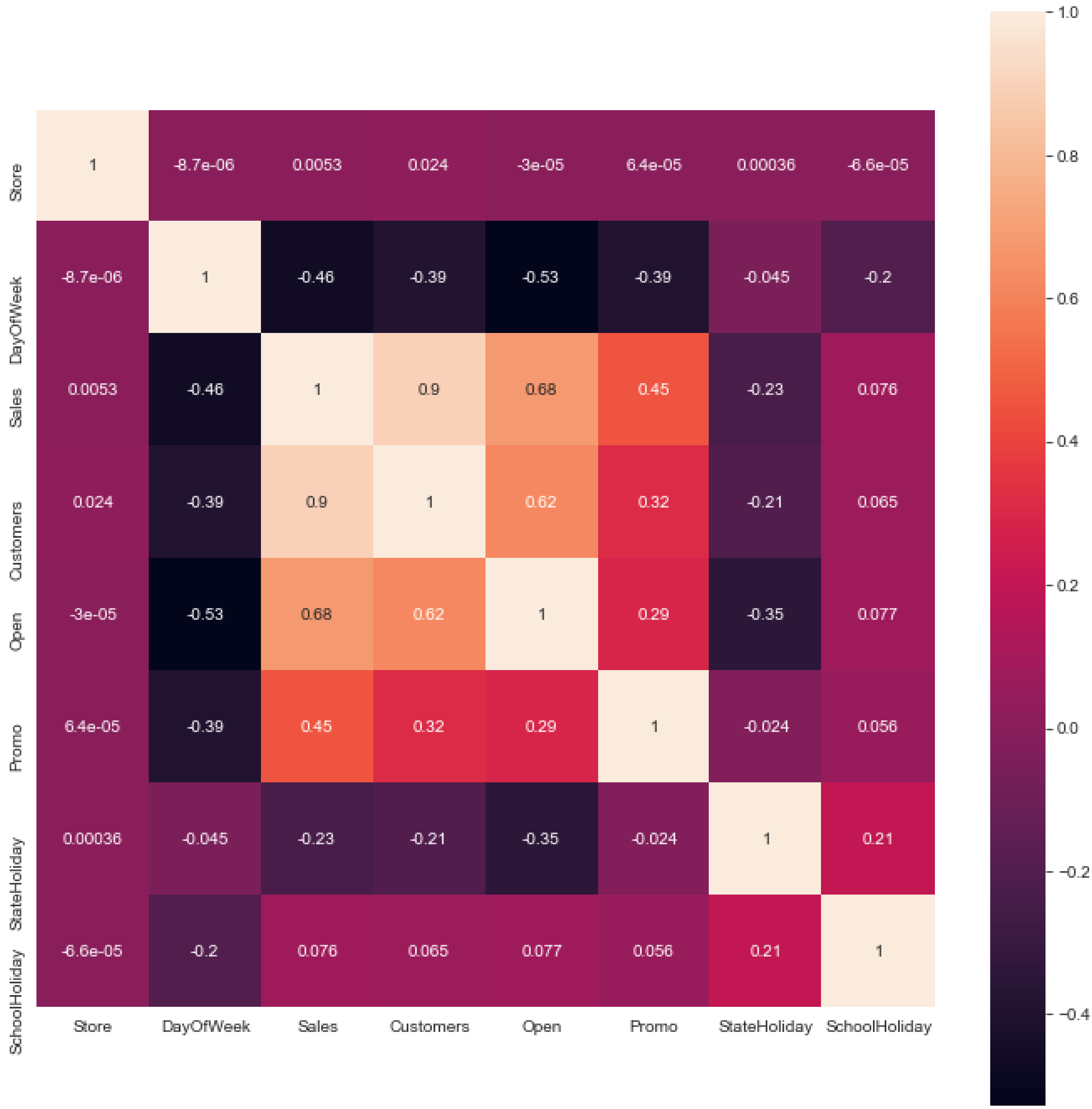
```
Store            0
DayOfWeek        0
Date             0
Sales            0
Customers        0
Open             0
Promo            0
StateHoliday     0
SchoolHoliday    0
dtype: int64
```

- There is no null value

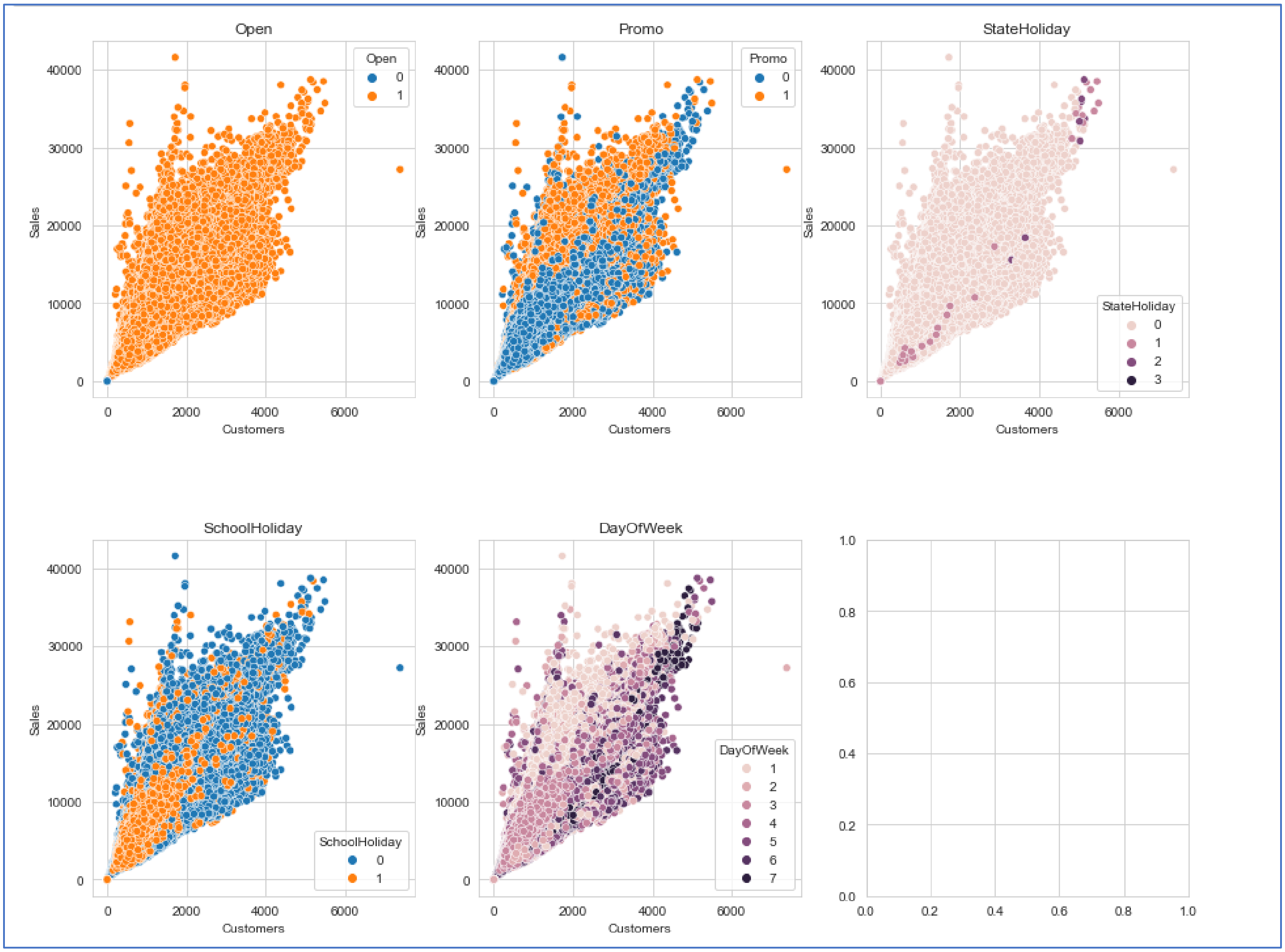
- Heat map of the correlation matrix of the complete dataset.

```python
# check the correlation between variables
plt.figure(figsize=(12,12))
sns.heatmap(train_data.corr(),annot=True, square=True)
```

```
<AxesSubplot:>
```



- Scatter plot of Sales vs Customer column.



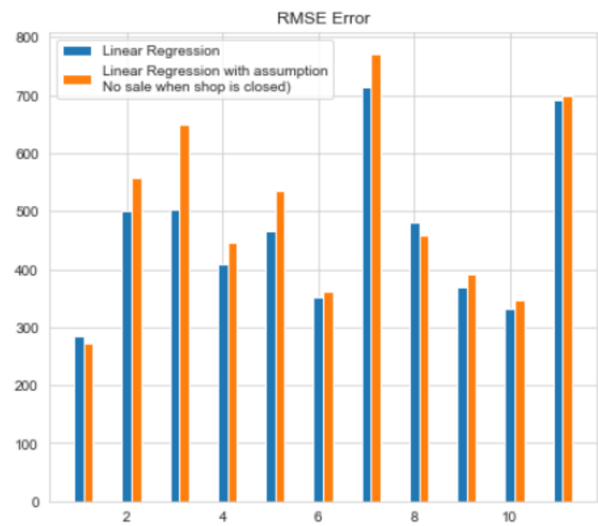
From the above EDA analysis we can conclude as follow:

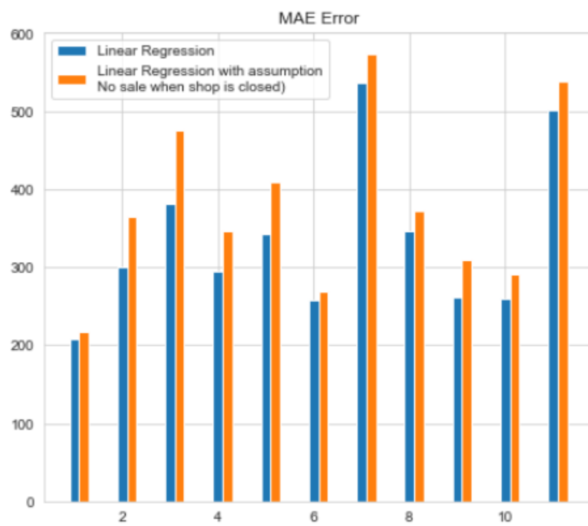
```
* Sales is zero when shop is closed.

* Sales is high when promo codes and discount is available.

* Sales is either very low or very high on State Holidays.

* Sales is high when schools are open.
```

2. Project Task: Week 2

Other Regression Techniques:

1. **When store is closed, sales = 0. Can this insight be used for Data Cleaning? Perform this and retrain the model. Any benefits of this step?**





- The above graph shoes that both types of error get increased when we removed the rows when store are closed.
- I think the main reason for this increased error rate is that, our previous model was predicting accurately when store was closed, so while taking mean of that portion the error output got reduced, but in updated model as we removed that rows, so while taking mean it get increased error output.
- So we do not get any benefit of removing those rows.

2. **Use Non-Linear Regressors like Random Forest or other Tree-based Regressors.**

   a) **Train a single model for all stores, where store Id can be a feature.**

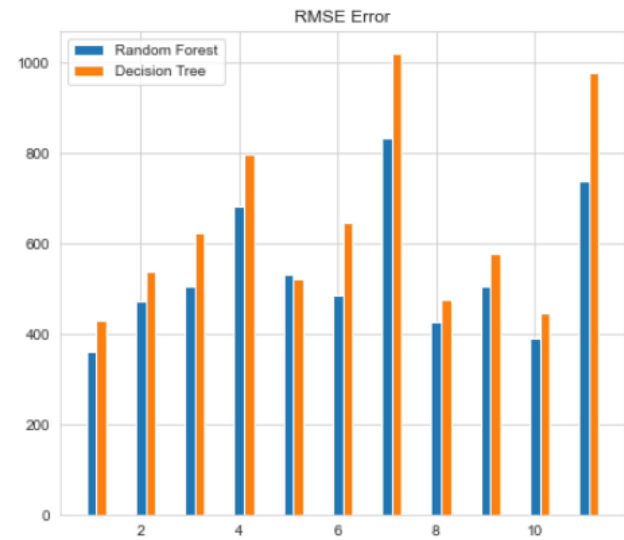
```
print('Root mean squared error: ',RMSE_rdm)
print('Mean absolute error: ',MAE_rdm)
```

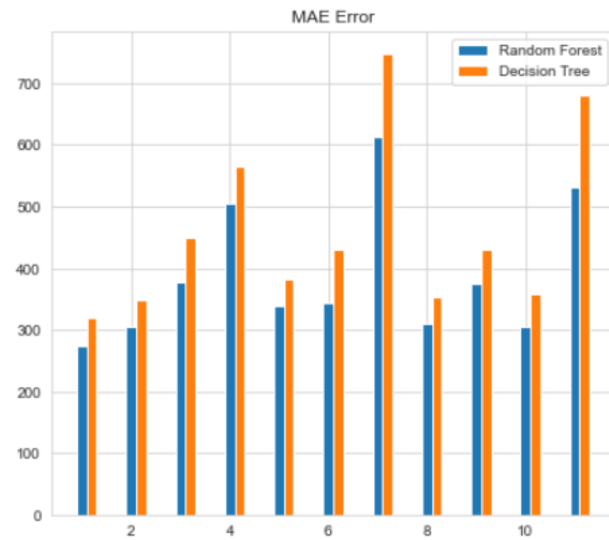
```
Root mean squared error:  734.5806667844583
Mean absolute error:  498.21204030910604
```

```
plt.figure(figsize=(16,8))
plt.plot(y_pred[:100],label = 'Sales forecast')
plt.plot(y_true[:100],label = 'Actual Sales')
plt.legend()
plt.title('Random Forest Regression taking all stores')
plt.show()
```

   b) **Train separate models for each store.**

	decision_tree	Random_forest
RMSE	1111.161370	934.724327
MAE	667.705973	604.585739

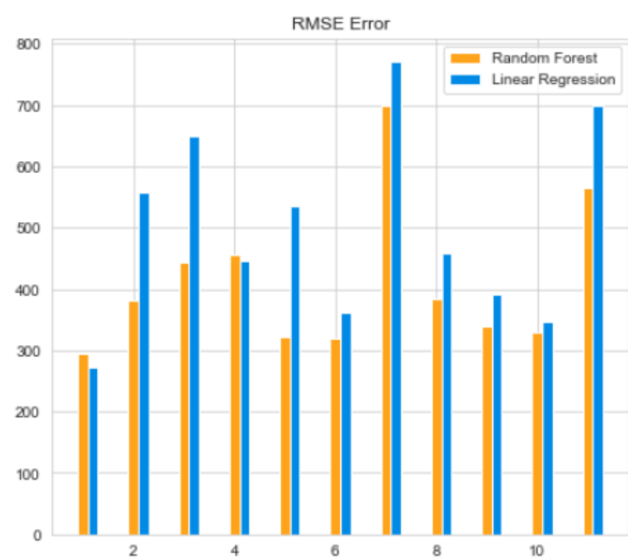


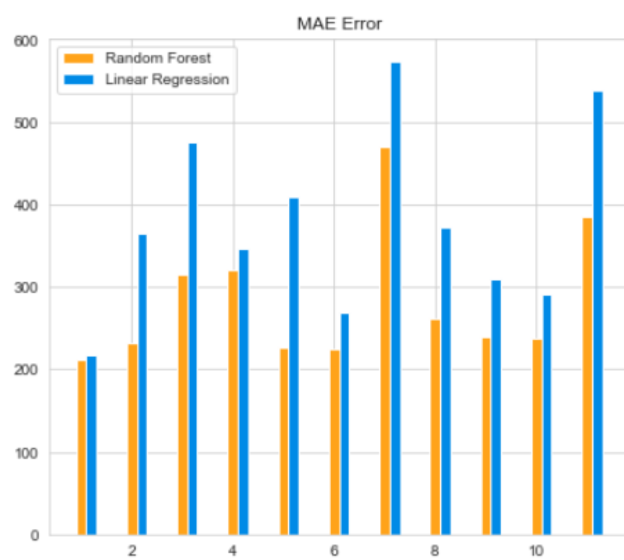



```
print('Average RMSE Decision Tree Error: {}'.format(error_output_dt_pca.RMSE.mean()))
print('Average RMSE Random Forest Error: {}'.format(error_output_rdm_pca.RMSE.mean()))
```

```
Average RMSE Decision Tree Error: 641.4061551719874
Average RMSE Random Forest Error: 539.3957964010076
```

3. Compare the performance of Linear Model and Non-Linear Model from the previous observations. Which performs better and why?





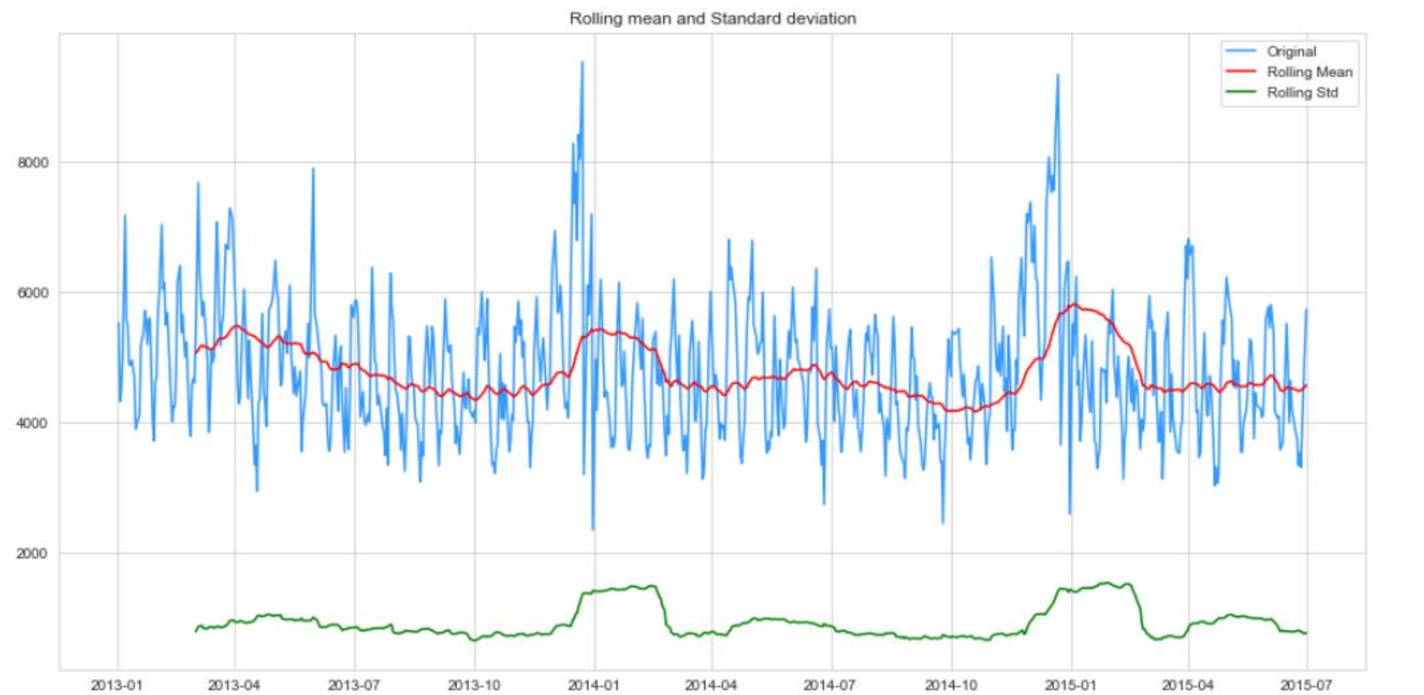

```
print('Average RMSE Random Forest Error: {}'.format(error_output_rdm.RMSE.mean()))
print('Average RMSE Linear Regression Error: {}'.format(error_output_lrc.RMSE.mean()))
```

```
Average RMSE Random Forest Error: 411.82760368071763
Average RMSE Linear Regression Error: 498.6368743223171
```

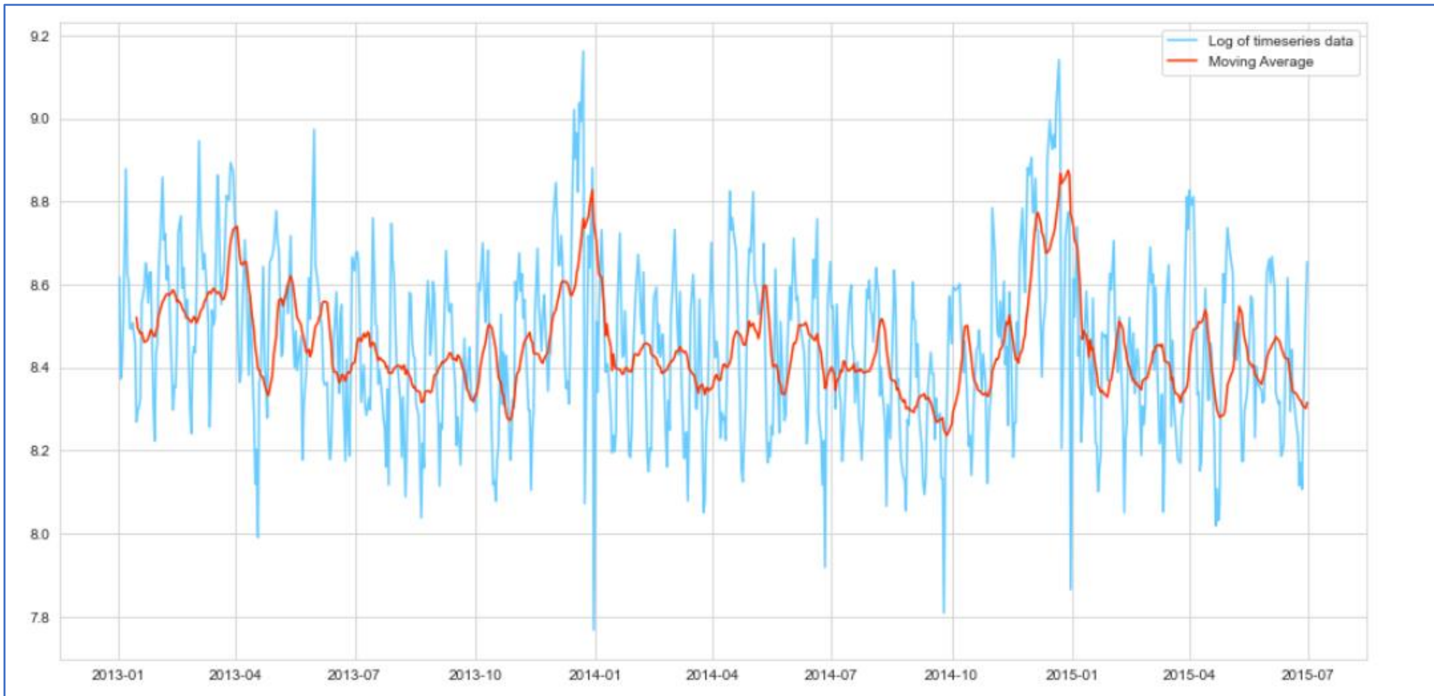
- From the above graph, it is clear that non-linear model i.e. random forest performs better than Linear Regression model.
- So we can say that our dataset is not lierly separable.

4. **Train a Time-series model on the data taking time as the only feature. This will be a store-level training.**

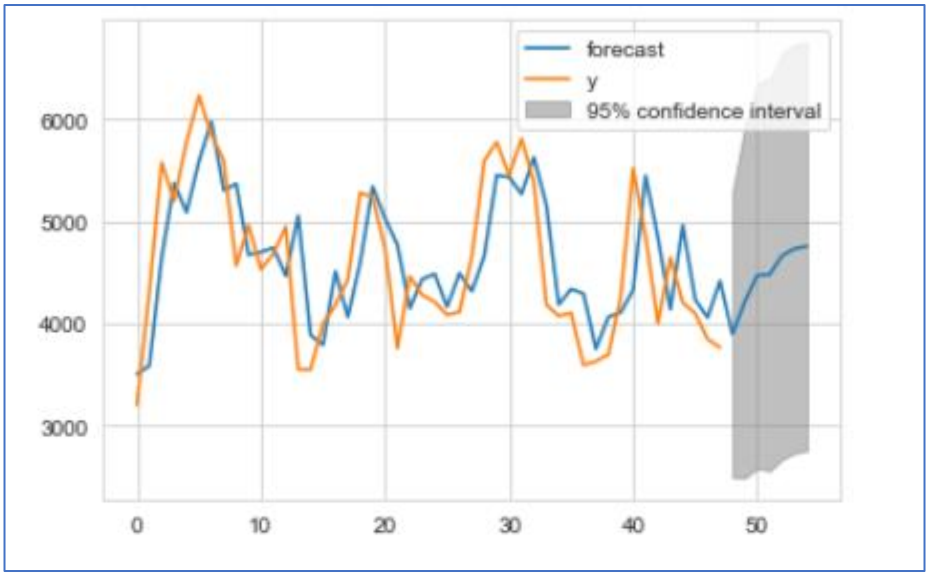
   a) Identify yearly trends and seasonal months




```
Result of Dickey-Fuller Test:
Test Statistic                 -5.336702
p-value                         0.000005
Number of lag used             13.000000
Number of observation used    740.000000
Critical Value (1%)            -3.439218
Critical Value (5%)            -2.865454
Critical Value (10%)           -2.568854
dtype: float64
```



- From the above graph we can see seasonal effect in the dataset.
- In dec to jan month sale is high in comaprison to other month




```
RMSE_ARIMA = math.sqrt(mean_squared_error(np.array(datax[700:]) , results.predict(700,753)))
RMSE_ARIMA
```

```
587.1520321281363
```

```
MAE_ARIMA = mean_absolute_error(np.array(datax[700:]) , results.predict(700,753))
MAE_ARIMA
```

```
482.5240578776619
```

- The above time-Series model is designed for Store ID one, similarly we can design for other Stores.

Project Task: Week 3

Implementing Neural Networks:

1. **Train a LSTM on the same set of features and compare the result with traditional time-series model.**

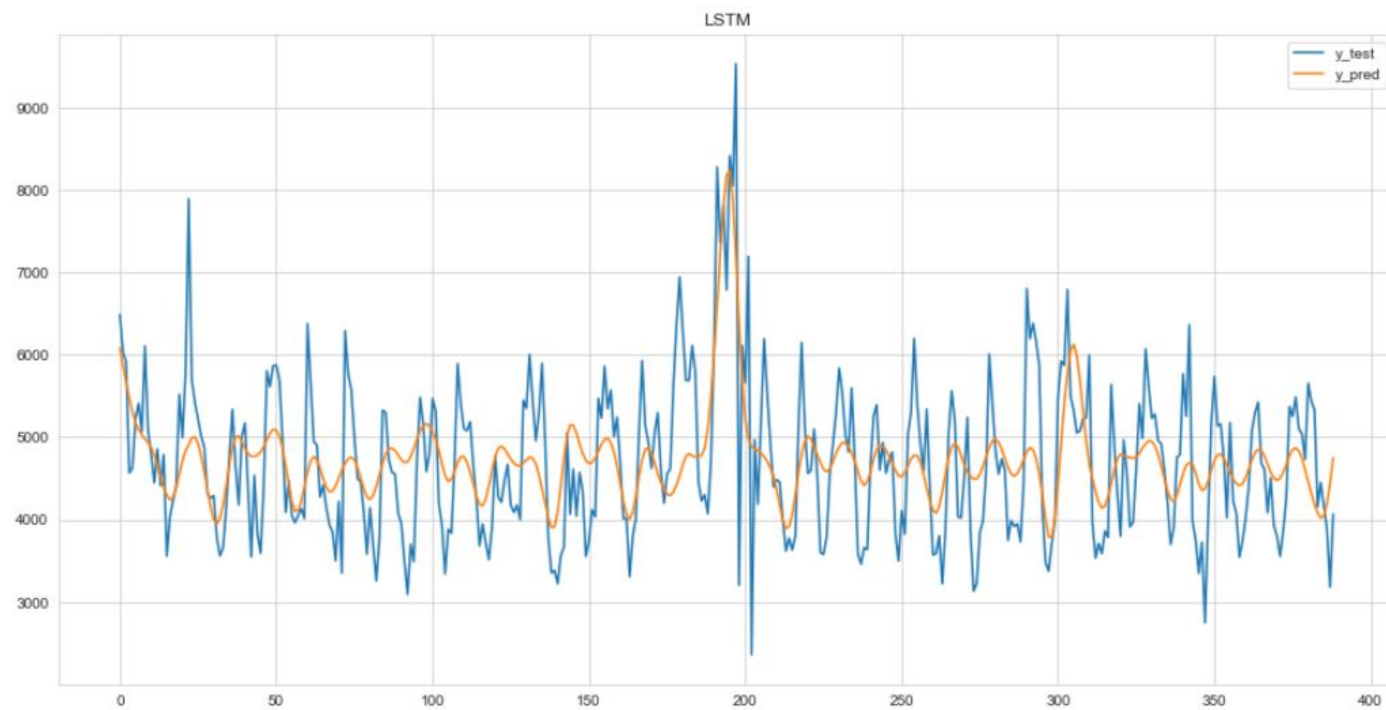
```
RMSE_LSTM = math.sqrt(mean_squared_error(y_train,train_predict))
RMSE_LSTM
```

```
758.7049250457853
```

```
RMSE_LSTM = math.sqrt(mean_squared_error(y_test,test_predict))
RMSE_LSTM
```

```
1038.4188839819349
```

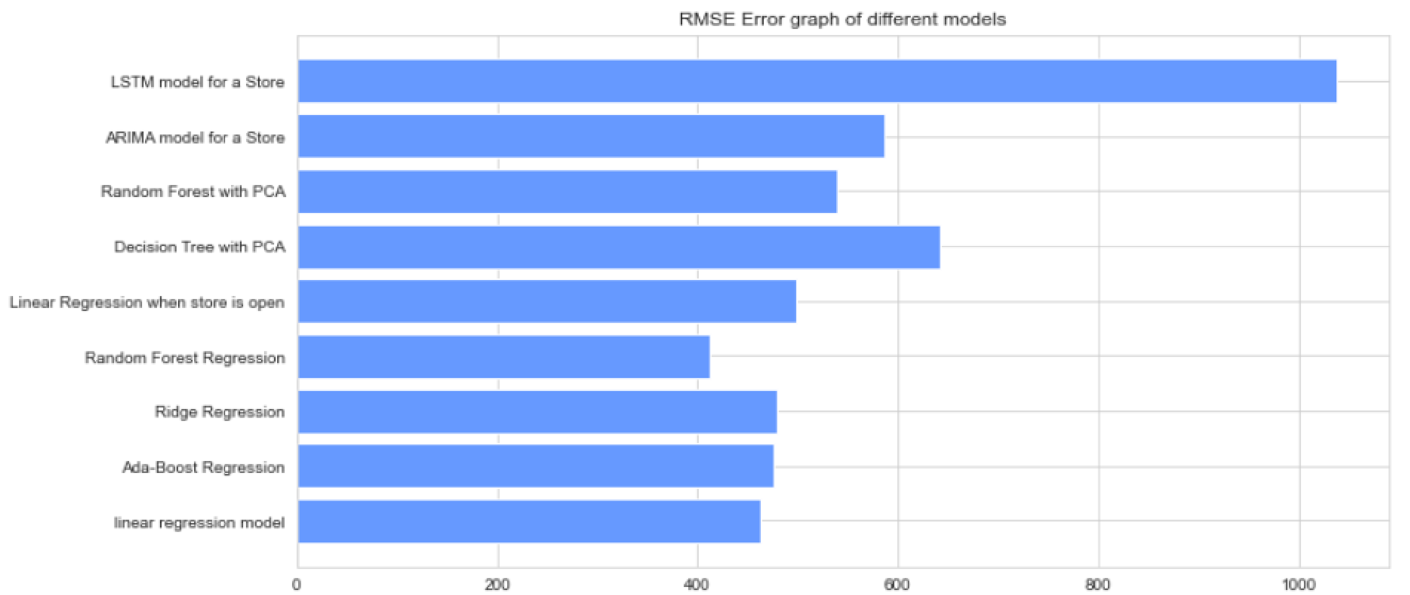
```
plt.figure(figsize = (16,8))
plt.plot(y_train, label = 'y_test')
plt.plot(train_predict,label = 'y_pred')
plt.title('LSTM')
plt.legend()
plt.show()
```



- Here, Tradional Time-Series models performs better than LSTM model.

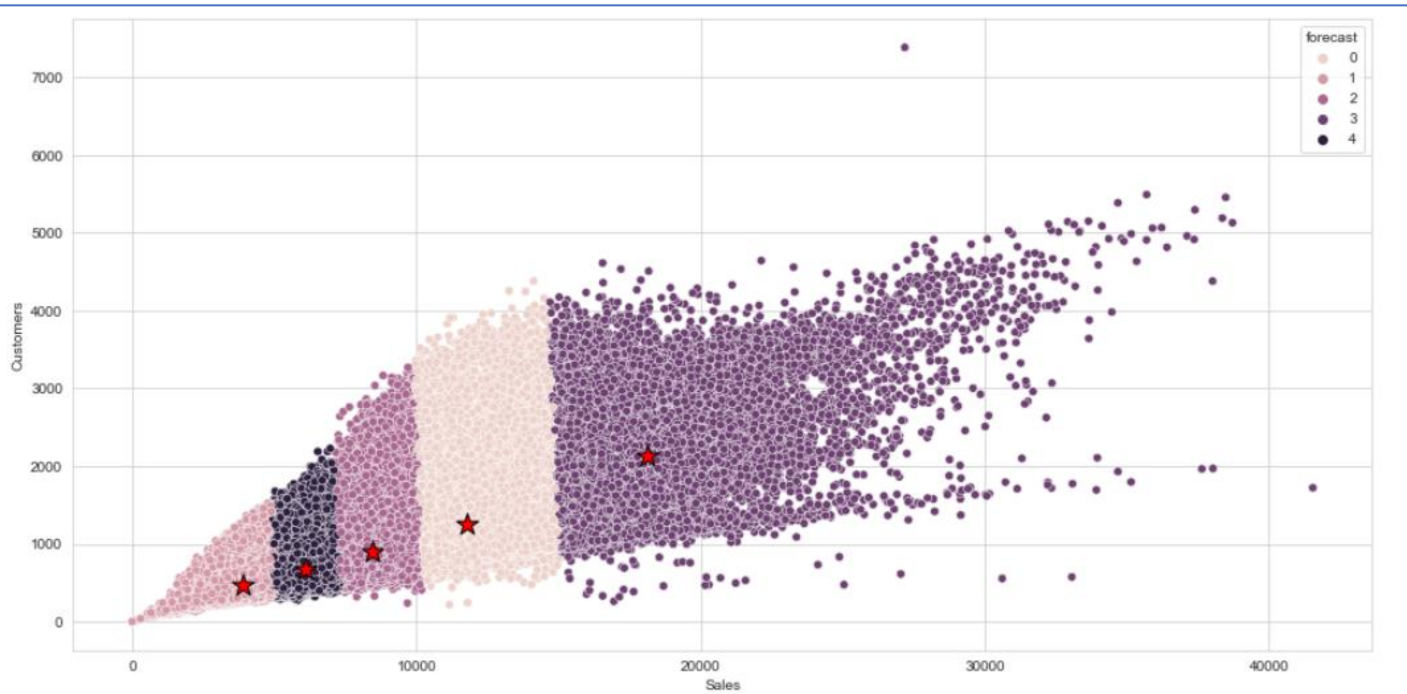
2. **Comment on the behavior of all the models you have built so far**

```
plt.figure(figsize=(12,6))
plt.barh(models_error[1],models_error[0], color = "#6699ff")
plt.title('RMSE Error graph of different models')
plt.show()
```

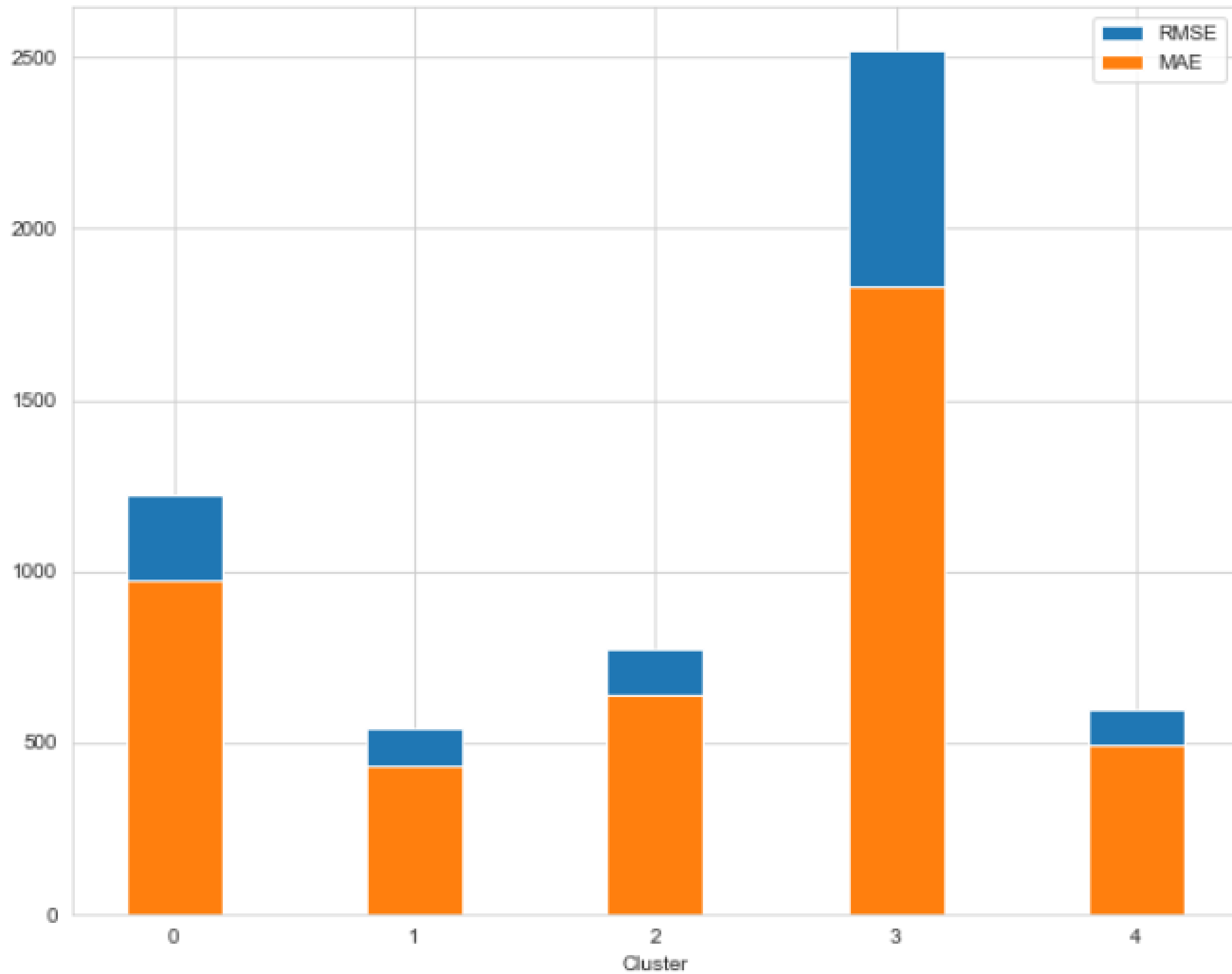


- From the above graph we can clearly says that Random forest performs best out of all models.

3. **Cluster stores using sales and customer visits as features. Find out how many clusters or groups are possible. Also visualize the results. – 5 clusters**



4. **Is it possible to have separate prediction models for each cluster? Compare results with the previous models.**



- since data is not suitable for clustring, we can not separate data into different clusters.
- so while predicting sales based on clusters, it shows unpredictible result (RMSE, and MAE)

➢ Project Task: Week 4

Applying ANN:

1. **Use ANN (Artificial Neural Network) to predict Store Sales.**
    - **a) Fine-tune number of layers,**
    - **b) Number of Neurons in each layers.**
    - **c) Experiment in batch-size.**
    - **d) Experiment with number of epochs. Carefully observe the loss and accuracy? What are the observations?**
    - **e) Play with different Learning Rate variants of Gradient Descent like Adam, SGD, RMS-prop.**
    - **f) Which activation performs best for this use case and why?**
    - **g) Check how it performed in the dataset, calculate RMSE.**

- I have tested so many combinations of hyper-parameters and finally i found best result as follow: -

```
model_1 = Sequential()
model_1.add(layers.Dense(32, activation='elu', input_shape = (x_train.shape[1],)))
model_1.add(layers.Dense(64, activation='elu'))
model_1.add(layers.Dense(64, activation='elu'))
model_1.add(layers.BatchNormalization())

## block 2
model_1.add(layers.Dense(128, activation='elu'))
model_1.add(layers.Dense(128, activation='elu'))
model_1.add(layers.BatchNormalization())

## block 3
model_1.add(layers.Dense(256, activation='elu'))
model_1.add(layers.Dense(256, activation='elu'))
model_1.add(layers.BatchNormalization())

## block 4
model_1.add(layers.Dense(128, activation='elu'))
model_1.add(layers.Dense(128, activation='elu'))
model_1.add(layers.BatchNormalization())

## block 5
model_1.add(layers.Dense(64, activation='elu'))
model_1.add(layers.Dense(64, activation='elu'))
model_1.add(layers.Dense(32, activation='elu'))
model_1.add(layers.Dense(1))
```

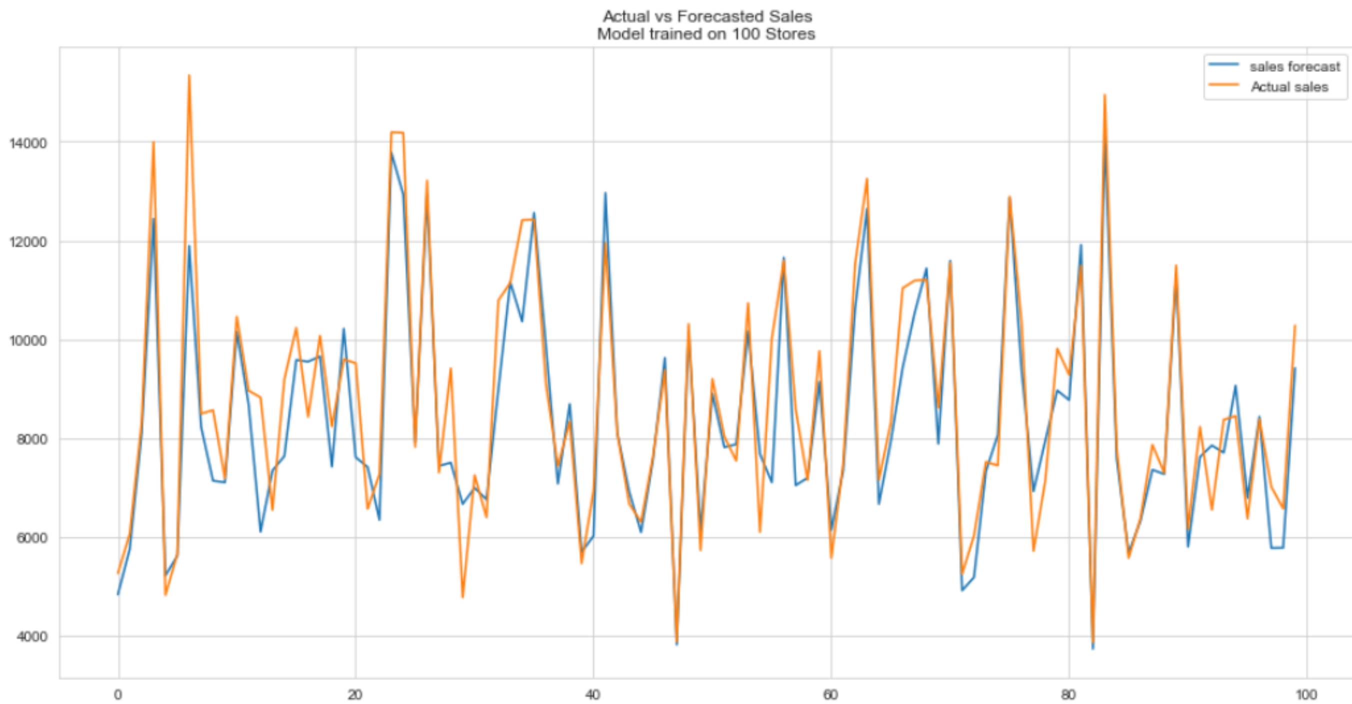
```
model_1.compile(loss='mse',
              optimizer = Adam(learning_rate=0.001),
              metrics=['mae'])
```

```
y_pred = model_1.predict(x)
math.sqrt(mean_squared_error(y,y_pred))
```

```
793.2148083439479
```

```
plt.figure(figsize=(16,8))
plt.plot(y_pred[:100],label = 'sales forecast')
plt.plot(y[:100],label = 'Actual sales')
plt.legend()
plt.title('Actual vs Forecasted Sales\nModel trained on 100 Stores')
```

```
Text(0.5, 1.0, 'Actual vs Forecasted Sales\nModel trained on 100 Stores')
```



2. **Use Dropout for ANN and find the optimum number of clusters (clusters formed considering the features: sales and customer visits). Compare model performance with traditional ML based prediction models.**

```
model_2 = load_model('Sales_ann_with_dropout.h5')

y_pred = model_2.predict(x_test)
math.sqrt(mean_squared_error(y_test,y_pred))
```

```
1938.6130502881572
```

- When I used dropout root mean squared error increased.
- It is not useful for our model.

3. **Find the best setting of neural net that minimizes the loss and can predict the sales best. Use techniques like Grid search, cross-validation and Random search.**

   - **K-Fold cross validation is performed below.**

```
score_kf_ann
```

```
[67055.82603188697, 1705.5901814265628, 5139.764923450266, 2165.0387761147335]
```

```
RMSE = sum(score_kf_ann)/len(score_kf_ann)
RMSE
```

```
19016.554978219632
```

- Prediction is quite good in comparision to mean.

- All points are covered successfully.