Predicting the daily demand of TV Models using Machine Learning

Salman Farshi
ID:1520162042
Salman.farshi@northsouth.edu

Introduction: Given data contains Jan-2014 to Aug-2016 daily TV sales quantity. There are total 124 Models. This data is collected from one of the leading brand of Bangladesh. Annually there are two big festivals (EID) which follows Islamic lunar calendar.We have predict the daily demand , how much tv daily need according to given data .

Data Collection: We got the data from Kaggle, data is well prepared, there are no missing data.

Features:

```
In [58]: df.head(10)
```

Out[58]:

	Date	Model	Count
0	31-Aug-16	M45	5
1	31-Aug-16	M121	3
2	31-Aug-16	M122	4
3	31-Aug-16	M91	10
4	31-Aug-16	M66	57
5	31-Aug-16	M100	59
6	31-Aug-16	M5	17
7	31-Aug-16	M14	653
8	31-Aug-16	M62	8
9	31-Aug-16	M15	25

```
In [60]: df.describe()
```

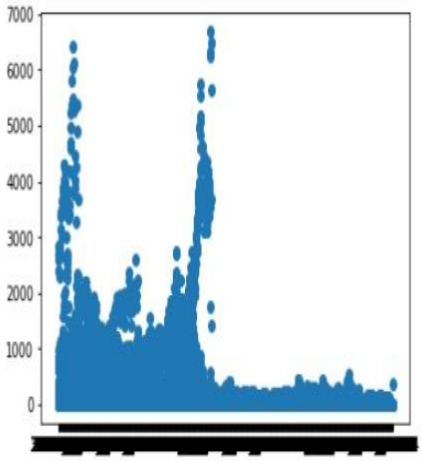
Out[60]:

	Count
count	46116.000000
mean	96.690108
std	315.065946
min	1.000000
25%	3.000000
50%	14.000000
75%	67.000000
max	6678.000000

Graph:

Date vs count plot before data parsing into date time object

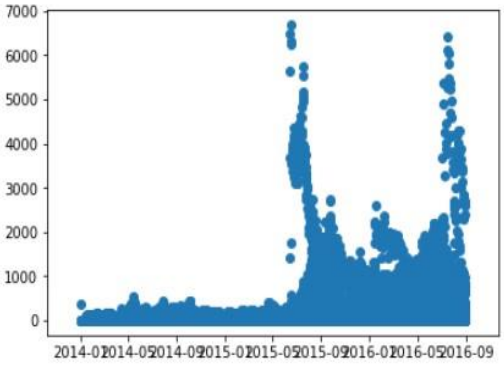
```
In [61]: fig, ax = plt.subplots()
         ax.scatter(df["Date"], df["Count"])

Out[61]: <matplotlib.collections.PathCollection at 0x7fa2eb8f9110>
```



Date vs count plot after data parsing into date time object

We can see that sales demand is increasing and during Eid it increase especially

```
In [69]: fig, ax = plt.subplots()
         ax.scatter(df["Date"], df["Count"])

Out[69]: <matplotlib.collections.PathCollection at 0x7fa2d1ff8990>
```



Divide the parsing data into day, month and year

```
In [72]: df_tmp = df.copy()

In [73]: df_tmp["saleYear"] = df_tmp.Date.dt.year
         df_tmp["saleMonth"] = df_tmp.Date.dt.month
         df_tmp["saleDay"] = df_tmp.Date.dt.day
         df_tmp["saleDayofweek"] = df_tmp.Date.dt.dayofweek
         df_tmp["saleDayofyear"] = df_tmp.Date.dt.dayofyear
         df_tmp.drop("Date", axis=1, inplace=True)

In [74]: df_tmp.head()
```

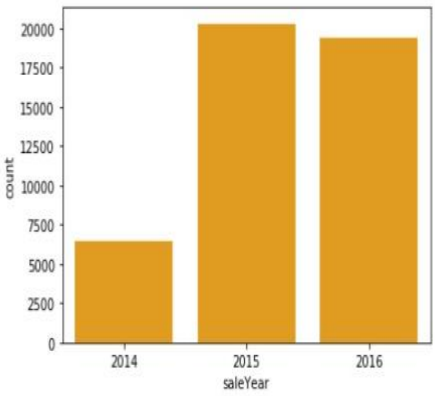
Out[74]:

	Model	Count	saleYear	saleMonth	saleDay	saleDayofweek	saleDayofyear
46115	M58	1	2014	1	1	2	1
46109	M88	2	2014	1	1	2	1
46110	M55	384	2014	1	1	2	1
46114	M54	1	2014	1	1	2	1
46112	M59	1	2014	1	1	2	1

Sale distribution based on year

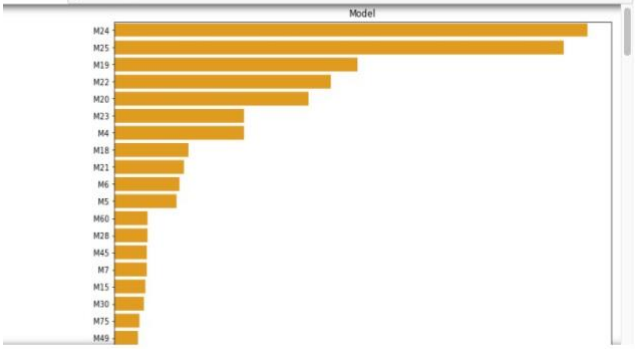
```
In [75]: sns.countplot(x='saleYear', data=df_tmp, color='orange')
```

Out[75]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa2d34c3e90>



The best model based on sale

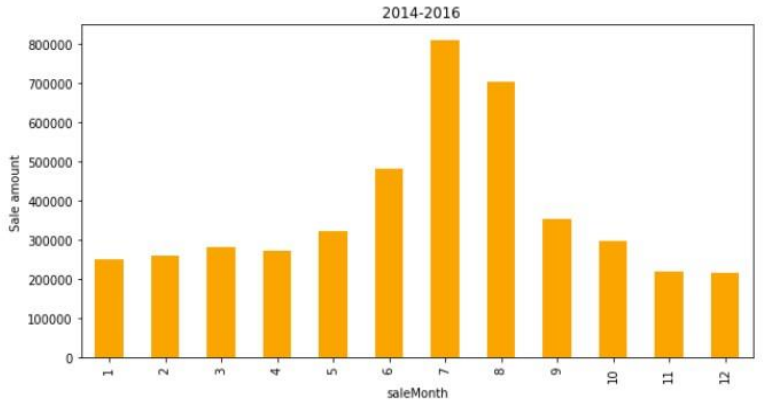
```
In [76]: models_by_sale = df.groupby('Model')['Count'].sum()
         models_by_sale.sort_values(axis=0, ascending=False, inplace=True)

         f, ax = plt.subplots(figsize=(12, 48))
         ax=sns.barplot(models_by_sale, models_by_sale.index,orient='h', color='Orange')
         ax.set(title='Model',xlabel='Count ', ylabel='Model name')
         plt.show()
```



Sale by month: Eid time sale demand increasing highly

```
In [78]: plt.figure(figsize=(10,5))
         plt.title('  2014-2016')
         plt.ylabel('Sale amount')
         df_tmp.groupby('saleMonth').Count.sum().plot(kind='bar',color='orange')
```

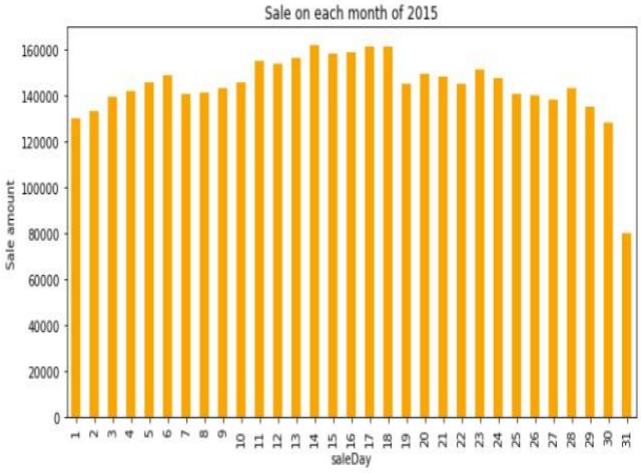
Out[78]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa2d353d3d0>



Sale by Day

In [80]:
```
plt.figure(figsize=(10,5))
plt.title(' Sale on each month of 2015 ')
plt.ylabel('Sale amount')
df_tmp.groupby('saleDay').Count.sum().plot( kind='bar',color='orange')
```

Out[80]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa2d2979950>



Most important Feature:

This is the time series problem, so data is the most import features and without date there is only Model features that is also import for predicting the result.

Correlation Matrix:

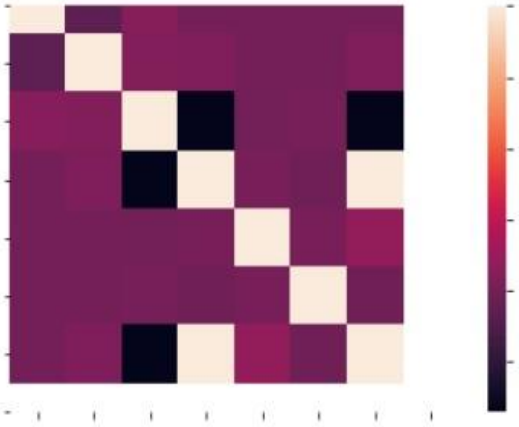
We have generated proper correlation matrix because its diagonal values is 1 . In this figure we cannot see the label annotation , I don' know the reason , I have done everything correctly

In [203]:
```
corr = df_tmp.corr()
corr['Count'].sort_values(ascending=False)
```

Out[203]:
```
Count             1.000000
saleYear          0.043856
saleMonth         0.034535
saleDayofyear     0.034261
saleDayofweek    -0.000921
saleDay          -0.001421
Model            -0.075704
Model_is_missing       NaN
Name: Count, dtype: float64
```

In [207]:
```
#plt.figure(figsize=(15,10))

sns.heatmap(df_tmp.corr(),annot=True)
```

Out[207]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa1fcf9ded0>
```



Regressor used: SVM , RandomForestRegressor, KNeighborsRegressor

Reasoning:

This data set is unknown to us and we are trying find out the best model which have good prediction ability with a great accuracy. That is why we have used different types of regressor models.

Model:

Random Forest Regressor without the best parameters

```
In [88]: from sklearn.ensemble import RandomForestRegressor
         model = RandomForestRegressor()
         model.fit(df_tmp.drop("Count", axis=1), df_tmp.Count)
```

```
/Users/salmanfarshi/opt/anaconda3/lib/python3.7/site-packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

```
Out[88]: RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                               max_features='auto', max_leaf_nodes=None,
                               min_impurity_decrease=0.0, min_impurity_split=None,
                               min_samples_leaf=1, min_samples_split=2,
                               min_weight_fraction_leaf=0.0, n_estimators=10,
                               n_jobs=None, oob_score=False, random_state=None,
                               verbose=0, warm_start=False)
```

Random Forest Regressor with the best parameters

```
In [192]: model = RandomForestRegressor(n_jobs=-1,
                                        random_state=42)
```

```
In [193]: model.fit(X_train, y_train)
```

```
/Users/salmanfarshi/opt/anaconda3/lib/python3.7/site-packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

```
Out[193]: RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                                max_features='auto', max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=-1,
                                oob_score=False, random_state=42, verbose=0,
                                warm_start=False)
```

KNeighborRegressor and Support vector Regressor

```
In [195]: model = KNeighborsRegressor()
          model.fit(X_train, y_train)
          show_scores(model)
```

In [196]:
```
model = SVR()
model.fit(X_train, y_train)
show_scores(model)
```

```
/Users/salmanfarshi/opt/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:19
3: FutureWarning: The default value of gamma will change from 'auto' to 'scale' in version
0.22 to account better for unscaled features. Set gamma explicitly to 'auto' or 'scale' to a
void this warning.
  "avoid this warning.", FutureWarning)
```

Model performance: accuracy

Random Forest with the best parameter

```
Out[194]: {'Training MAE': 5.4421746037896686,
           'Test MAE': 8.030945,
           'Training RMSLE': 0.24296706016275885,
           'Test RMSLE': 0.4109093860636579,
           'Training R^2': 0.9933338804973311,
           'Test R^2': 0.9847891033414012}
```

KNeighborsRegressor

```
In [195]: model = KNeighborsRegressor()
          model.fit(X_train, y_train)
          show_scores(model)

Out[195]: {'Training MAE': 50.67534389894215,
           'Test MAE': 59.86277,
           'Training RMSLE': 1.2737644573412887,
           'Test RMSLE': 1.3734938920454094,
           'Training R^2': 0.7756939856601988,
           'Test R^2': 0.7692460870413595}
```

Support vector Regressor

```
In [196]: model = SVR()
          model.fit(X_train, y_train)
          show_scores(model)

          /Users/salmanfarshi/opt/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:19
          3: FutureWarning: The default value of gamma will change from 'auto' to 'scale' in version
          0.22 to account better for unscaled features. Set gamma explicitly to 'auto' or 'scale' to
          void this warning.
            "avoid this warning.", FutureWarning)

Out[196]: {'Training MAE': 87.55015427356942,
           'Test MAE': 100.46195701784595,
           'Training RMSLE': 1.6121244252756295,
           'Test RMSLE': 1.6882073401251878,
           'Training R^2': -0.0629757994972342,
           'Test R^2': -0.0653734760628075 2}
```

Model Evolution: MAE,MSE and RMSE :

Mean Squared **Error** (**MSE**) and Root **Mean Square Error** penalizes the large prediction **errors** vi-a-vis Mean Absolute **Error** (**MAE**). ... **MAE** is more robust to data with outliers. The lower value of **MAE**, **MSE**, and RMSE implies higher accuracy of a regression model. However, a higher value of R square is considered desirable

We have trained our data with different types of models and we found our desirable model which can predict more correctly. We can see that the lowest MAE and RMSE are found in random forest regressor with the best parameter. It indicates that our data is distributed as such a way for which random forest is the best model. In random forest the obtained.

```
'Training MAE': 5.4421746037896686,
'Test MAE': 8.030945,
'Training RMSLE': 0.24296706016275885,
'Test RMSLE': 0.4109093860636579,
'Training R^2': 0.9933338804973311,
'Test R^2': 0.9847891033414012
```

We can observe here that the difference between the training errors and test errors are not so big. That indicates that our random forest model does not hugely overfitted.

Conclusion: In the last, we can say that random forest with the best parameters may be the best this sort of time series data . We have faced different types obstacle for the time series data but in the end of the day we find out. In future this model should be tested on real data then we can decide how to deal with future work. Due to short time we cannot use a lot of graphical figure which can show the project better way .