Product sales forecasting

Approach of solving this problem

Name – Shivani Sadani

This project focuses on developing a predictive model that uses historical sales data from different stores to forecast sales for upcoming periods.

Data Description

Train Data

Rows: 188340, Columns: 10
Y Variable: Sales

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 188340 entries, 0 to 188339
Data columns (total 10 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   ID             188340 non-null  object
 1   Store_id       188340 non-null  int64
 2   Store_Type     188340 non-null  object
 3   Location_Type  188340 non-null  object
 4   Region_Code    188340 non-null  object
 5   Date           188340 non-null  object
 6   Holiday        188340 non-null  int64
 7   Discount       188340 non-null  object
 8   #Order         188340 non-null  int64
 9   Sales          188340 non-null  float64
dtypes: float64(1), int64(3), object(6)
memory usage: 14.4+ MB
```

	Store_id	Holiday	#Order	Sales
count	188340.000000	188340.000000	188340.000000	188340.000000
mean	183.000000	0.131783	68.205692	42784.327982
std	105.366308	0.338256	30.467415	18456.708302
min	1.000000	0.000000	0.000000	0.000000
25%	92.000000	0.000000	48.000000	30426.000000
50%	183.000000	0.000000	63.000000	39678.000000
75%	274.000000	0.000000	82.000000	51909.000000
max	365.000000	1.000000	371.000000	247215.000000

	ID	Store_Type	Location_Type	Region_Code	Date	Discount
count	188340	188340	188340	188340	188340	188340
unique	188340	4	5	4	516	2
top	T1000001	S1	L1	R1	2018-01-01	No
freq	1	88752	85140	63984	365	104051

Steps in Building ML Model

1 Reading Data

2 Data Preprocessing
- Handling Missing Values
- Handling Outliers

3 Exploratory Data Analysis
- Exploring data to reveal hidden insights

4 Feature Engineering
- Generating New Features
- Categorical Encodings

5 Setting up Validation Strategy
- Splitting train & validation set based on date

6 Feature Scaling
- MinMax Scaler

7 Model Building

8 Cross Validation
- To make model robust over fitting

Data Preprocessing | Exploratory Data Analysis | Feature Engineering | Feature Scaling | Validation Strategy | Model Building & Comparison

Checking for Missing Values

- No missing values in Dataset

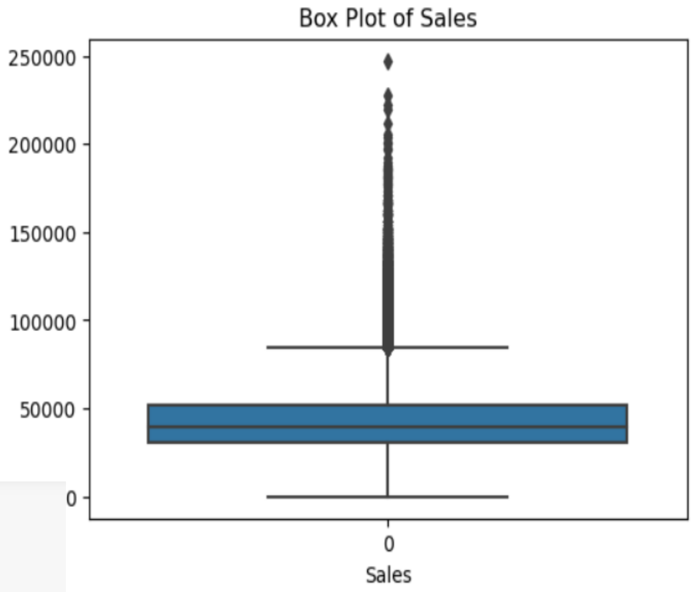
```
train_data.isnull().sum()
```

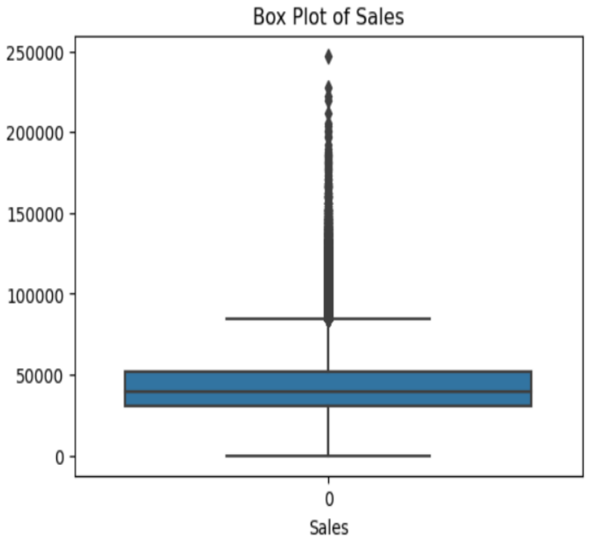
```
ID               0
Store_id         0
Store_Type       0
Location_Type    0
Region_Code      0
Date             0
Holiday          0
Discount         0
#Order           0
Sales            0
dtype: int64
```

```
#Handle Outliers

# Calculate Q1, Q3, and IQR
q1 = train['Sales'].quantile(0.25)
q3 = train['Sales'].quantile(0.75)
iqr = q3 - q1

# Calculate outlier thresholds
Lower_tail = q1 - 1.5 * iqr
Upper_tail = q3 + 1.5 * iqr

# Calculate the median
med = np.median(train['Sales'])

# Replace outliers with the median
train['Sales'] = np.where((train['Sales'] < Lower_tail) | (train['Sales'] > Upper_tail), med, train['Sales'])
```

Handling Outliers

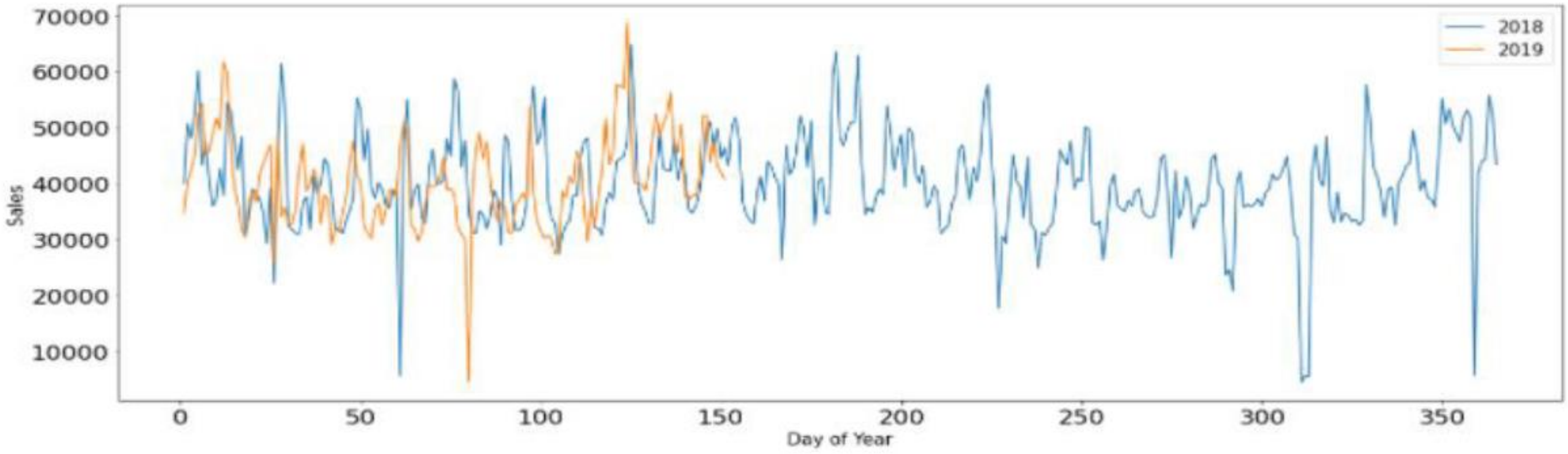
- Sales & Order column contain outliers
- Can’t remove orders’ outliers since these data points reveal important information.
- And outliers of sales column is being replaced by median.



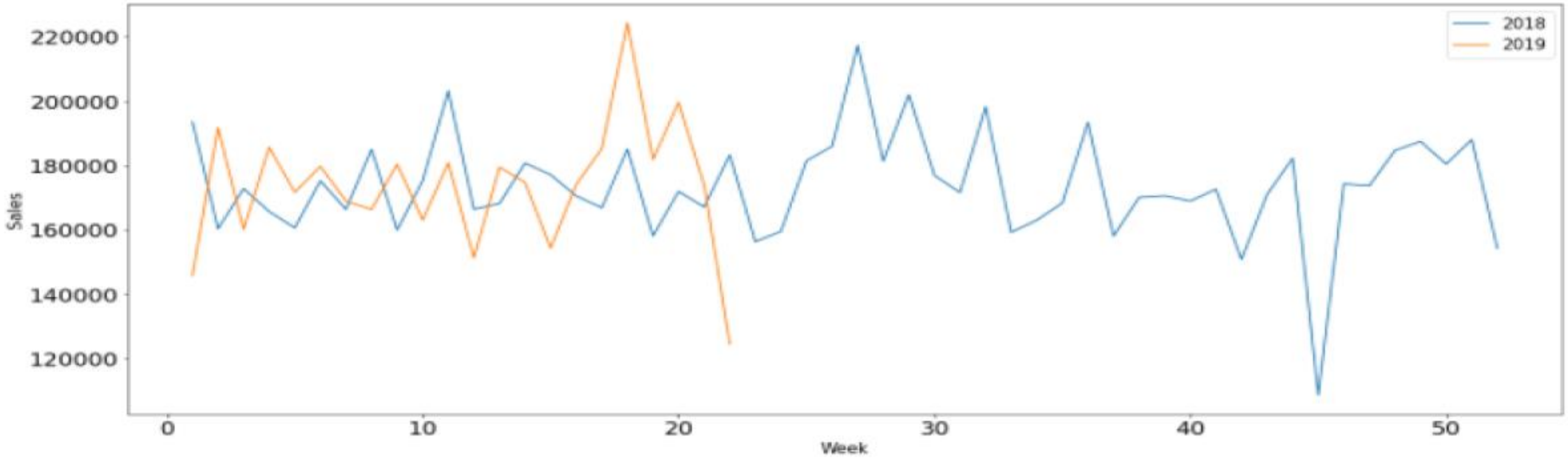


Data Preprocessing | Exploratory Data Analysis | Feature Engineering | Feature Scaling | Validation Strategy | Model Building & Comparison

Median Daily Sales



Median Weekly Sales



Data Preprocessing → Exploratory Data Analysis → Feature Engineering → Feature Scaling → Validation Strategy → Model Building & Comparison

New Features

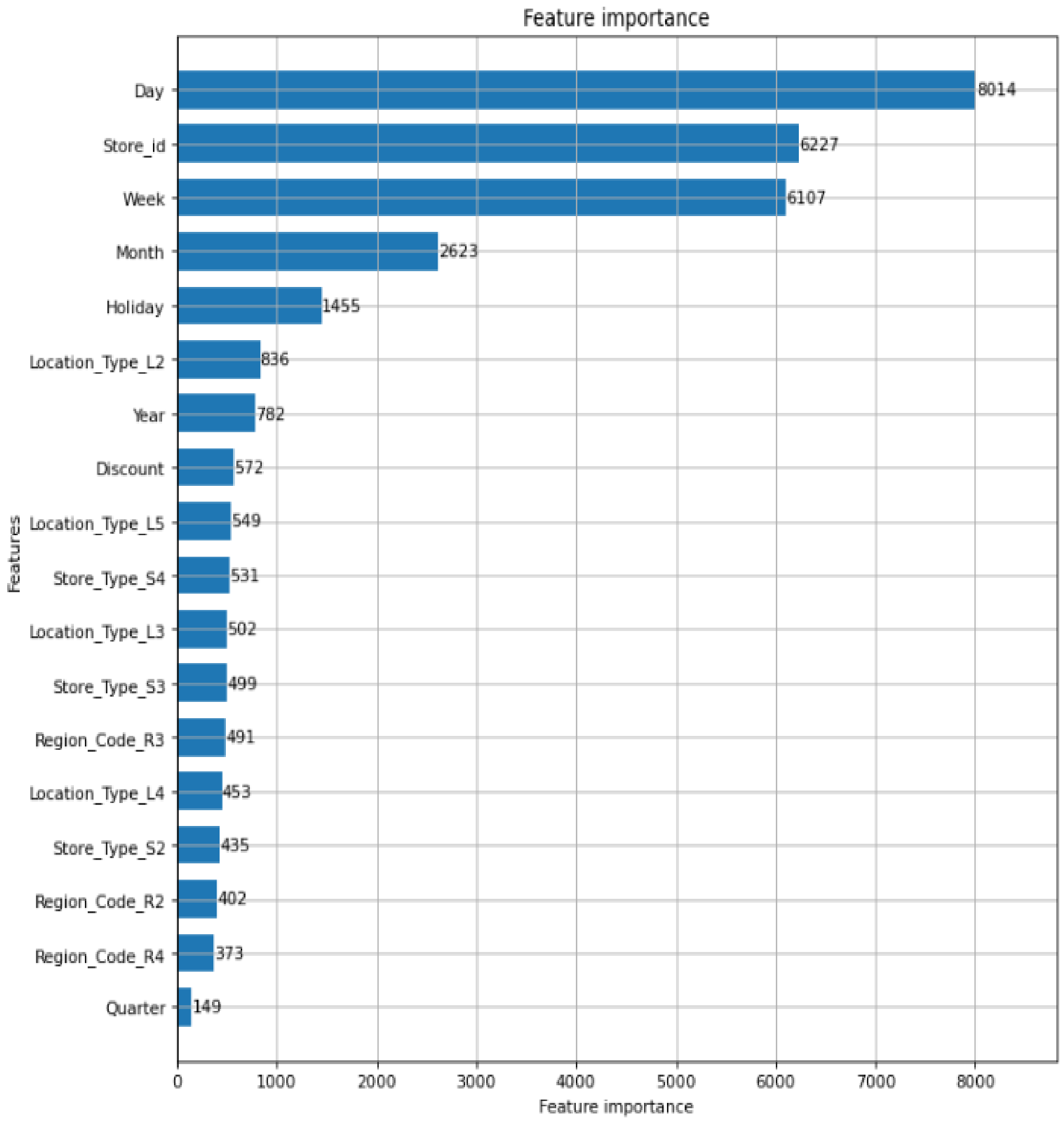
- Day
- Week
- Month
- Quarter

Feature Encoding

- One Hot Encoding for nominal categorical variables

```
# One Hot Encoding
l = ['Store_Type','Location_Type','Region_Code']
tr_x = pd.get_dummies(tr_x, columns = l,drop_first=True)
val_x = pd.get_dummies(val_x, columns = l,drop_first=True)
test = pd.get_dummies(test, columns = l,drop_first=True)
```

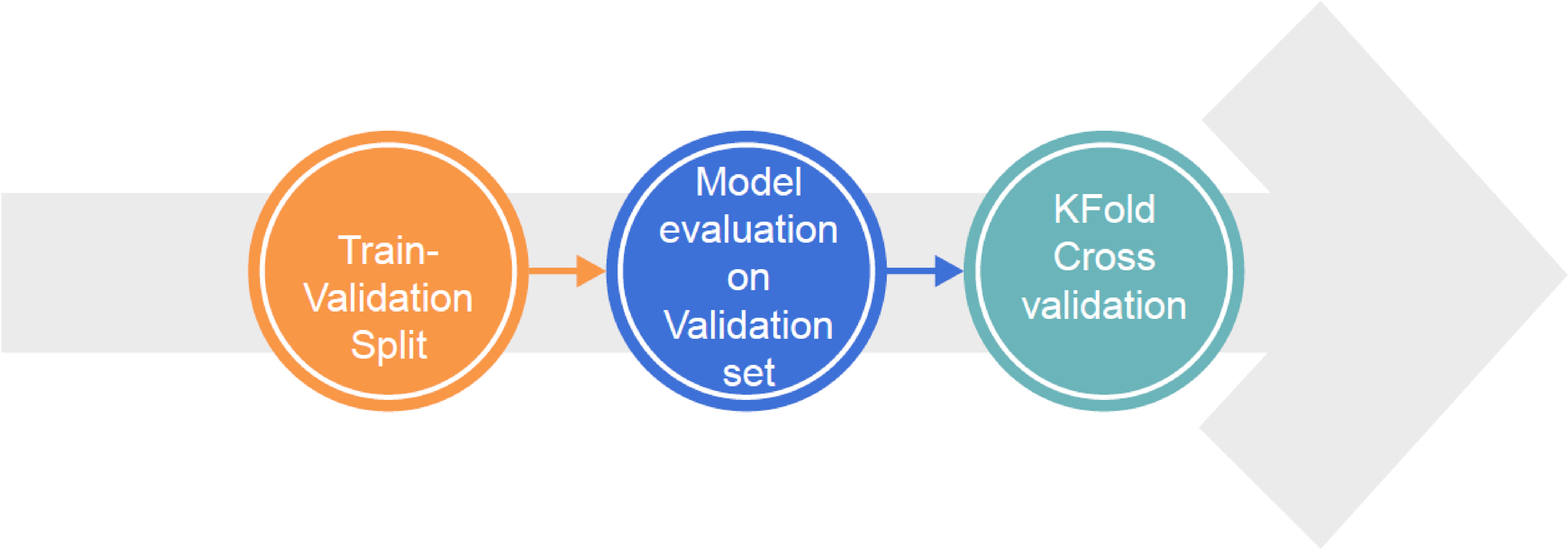
```
tr_x['Discount'] = tr_x['Discount'].replace({'Yes':1,'No':0})
val_x['Discount'] = val_x['Discount'].replace({'Yes':1,'No':0})
test['Discount'] = test['Discount'].replace({'Yes':1,'No':0})
```



Data Preprocessing | Exploratory Data Analysis | Feature Engineering | Feature Scaling | Validation Strategy | Model Building & Comparison

MinMax Scaler for Numeric Columns

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
scaler.fit(tr_x[numcols])
tr_x[numcols] = scaler.transform(tr_x[numcols])
val_x[numcols] = scaler.transform(val_x[numcols])
test[numcols] = scaler.transform(test[numcols])
```



- Data Before 1st April 2019 : Train Set
- Data from 1st April 2019: Validation Set

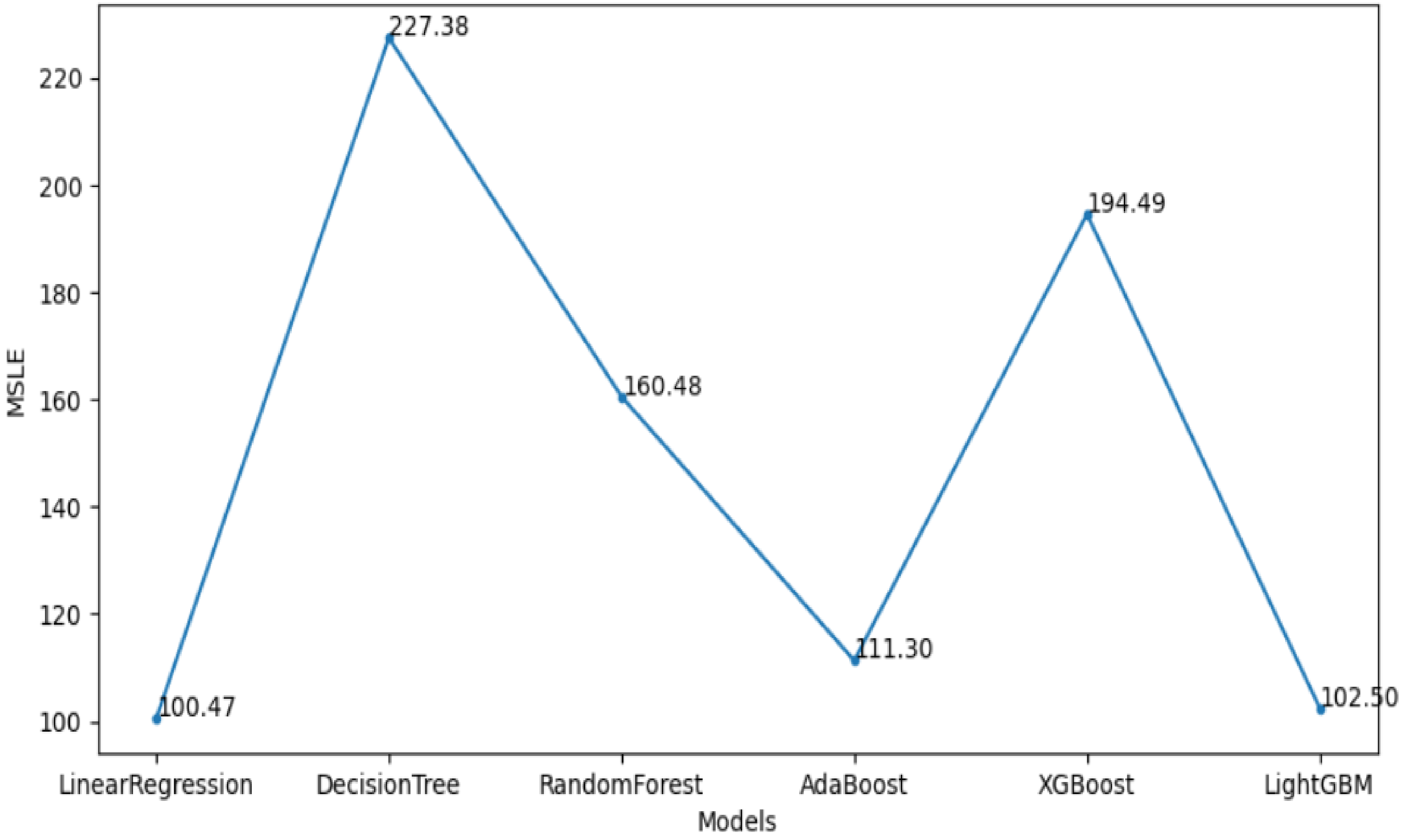
Building models and tuning hyper parameters on train set and evaluating on validation set.

KFold cross validation on train set for final model evaluation.

Data Preprocessing → Exploratory Data Analysis → Feature Engineering → Feature Scaling → Validation Strategy → Model Building & Comparison

Total Models Built: 6

Linear Regression	Decision Tree	Random Forest	AdaBoost	XgBoost	LightGBM



- Best performing model is LightGBM

	Models	MSLE
0	LinearRegression	100.474017
1	DecisionTree	227.379557
2	RandomForest	160.476955
3	AdaBoost	111.297497
4	XGBoost	194.490098
5	LightGBM	102.501614

After Kfold Cross validation it is found that LightGBM scores is better because both the validation and training MSLE values are lower i.e. 64.6.