CODING DOJO

الأكاديمية السعودية الرقمية
SAUDI DIGITAL ACADEMY

مجموعة تداول السعودية
Saudi Tadawul Group

SDA Capstone Project

Applying Machine Learning on Saudi Stock Exchange (Tadawul)

Data Divers Group

Zahra Almahd

Razan Aljohani

Fatima Alramadan

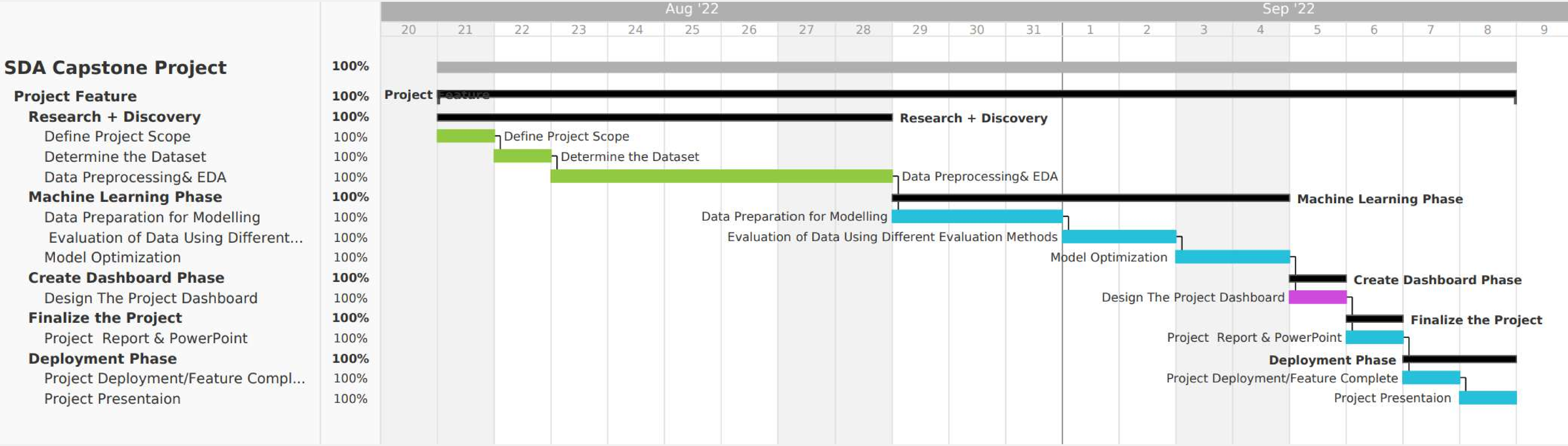
Asma Alasmary

Lena Baeshen

Content

1. Introduction (Tadawul & 2030 Vision)
2. Preprocessing Data
3. Saudi Stock Exchange Dashboard (Power BI)
4. Data Visualization
5. Machine Learning models
6. Models Optimization
7. Models Pipeline

Gantt Chart



Aug '22
Sep '22
20 21 22 23 24 25 26 27 28 29 30 31 1 2 3 4 5 6 7 8 9
SDA Capstone Project 100%
Project Feature 100%
Research + Discovery 100%
Define Project Scope 100%
Determine the Dataset 100%
Data Preprocessing& EDA 100%
Machine Learning Phase 100%
Data Preparation for Modelling 100%
Evaluation of Data Using Different... 100%
Model Optimization 100%
Create Dashboard Phase 100%
Design The Project Dashboard 100%
Finalize the Project 100%
Project Report & PowerPoint 100%
Deployment Phase 100%
Project Deployment/Feature Compl... 100%
Project Presentaion 100%
Evaluation of Data Using Different Evaluation Methods
Project Deployment/Feature Complete

Tadawul & 2030 Vision

Tadawul & 2030 Vision

To raise the private sector's contribution to GDP, from 40% to 65%.

To increase the Public Investment Fund's assets from SAR 600 Bn to over SAR 7 Tn.

To rise from our current position of 25 to the top 10 countries on the Global Competitiveness Index.

To move from our current position as the 19th largest economy in the world to the top 15.

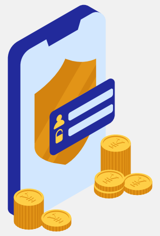
VISION رؤية 2030 المملكة العربية السعودية KINGDOM OF SAUDI ARABIA

1. Saudi Stock Exchange (Tadawul) Dataset

About Saudi Stock Exchange (TADAWUL)

The Saudi Exchange is a fully owned subsidiary by Saudi Tadawul Group

It carries out listing and trading in securities for local and international investors



Saudi Stock Exchange is providing long-term growth plans for the Group.

Dataset Information

- We did use the Saudi Stock Exchange (Tadawul) dataset from kaggle website.
- The dataset has 14 features and 593820 records.
- Each row in the database represents the price of a specific stock at a specific date:

  - symbol (Integer): The symbol or the reference number of the company
  - name(String) Name of the company
  - trading_name (String): The trading name of the company
  - sector (String): The sector in which the company operates
  - date (Date): The date of the stock price

Introduction

- open (Decimal): The opening price
- high (Decimal): The highest price of the stock at that day
- low (Decimal): The lowest price of the stock at that day
- close (Decimal): The closing price
- change (Decimal): The change in price from the last day
- perc_Change (Decimal): The percentage of the change
- volume_traded (Decimal): The volume of the trades for the day
- value_traded (Decimal): The value of the trades for the day
- no_trades (Decimal): The number of trades for the day

name	trading_name	sectoer	date	open	high	low	close	change	perc_Change	volume_traded	value_traded	no_trades
Saudi Arabia Refineries Co.	SARCO	Energy	2020-03-05	35.55	35.85	34.90	34.90	-0.40	-1.13	436609.0	15399073.50	804.0
Saudi Arabia Refineries Co.	SARCO	Energy	2020-03-04	34.70	35.65	34.50	35.30	0.25	0.71	737624.0	25981391.35	1268.0
Saudi Arabia Refineries Co.	SARCO	Energy	2020-03-03	34.70	35.15	34.70	35.05	1.05	3.09	489831.0	17116413.40	854.0

2. Preprocessing Data

Preprcessing

Missing Values

The dataset has missing values and has been deleted

```
#Check null value
df.isnull().sum()
```

```
symbol              0
name                0
trading_name        0
sectoer             0
date                0
open             4650
high             4650
low              4650
close               0
change              0
perc_Change         0
volume_traded       0
value_traded        0
no_trades        5084
dtype: int64
```

```
#drop null value
df=df.dropna()
```

Duplicated Records

The dataset doesn't have any duplicate records

```
df.duplicated().any()
```

```
False
```

Preprcessing

Deleting unnecessary columns

We deleted “Symbol” and “name” columns

```
In [15]: df=df.drop(['symbol','name'],axis=1)

In [16]: df.isnull().sum()

Out[16]: trading_name     0
         sectoer          0
         date             0
         open             0
         high             0
         low              0
         close            0
         change           0
         perc_Change      0
         volume_traded    0
         value_traded     0
         no_trades        0
         dtype: int64
```

Adding new columns

We did split “Date” column to new two columns “Year” and “Month”

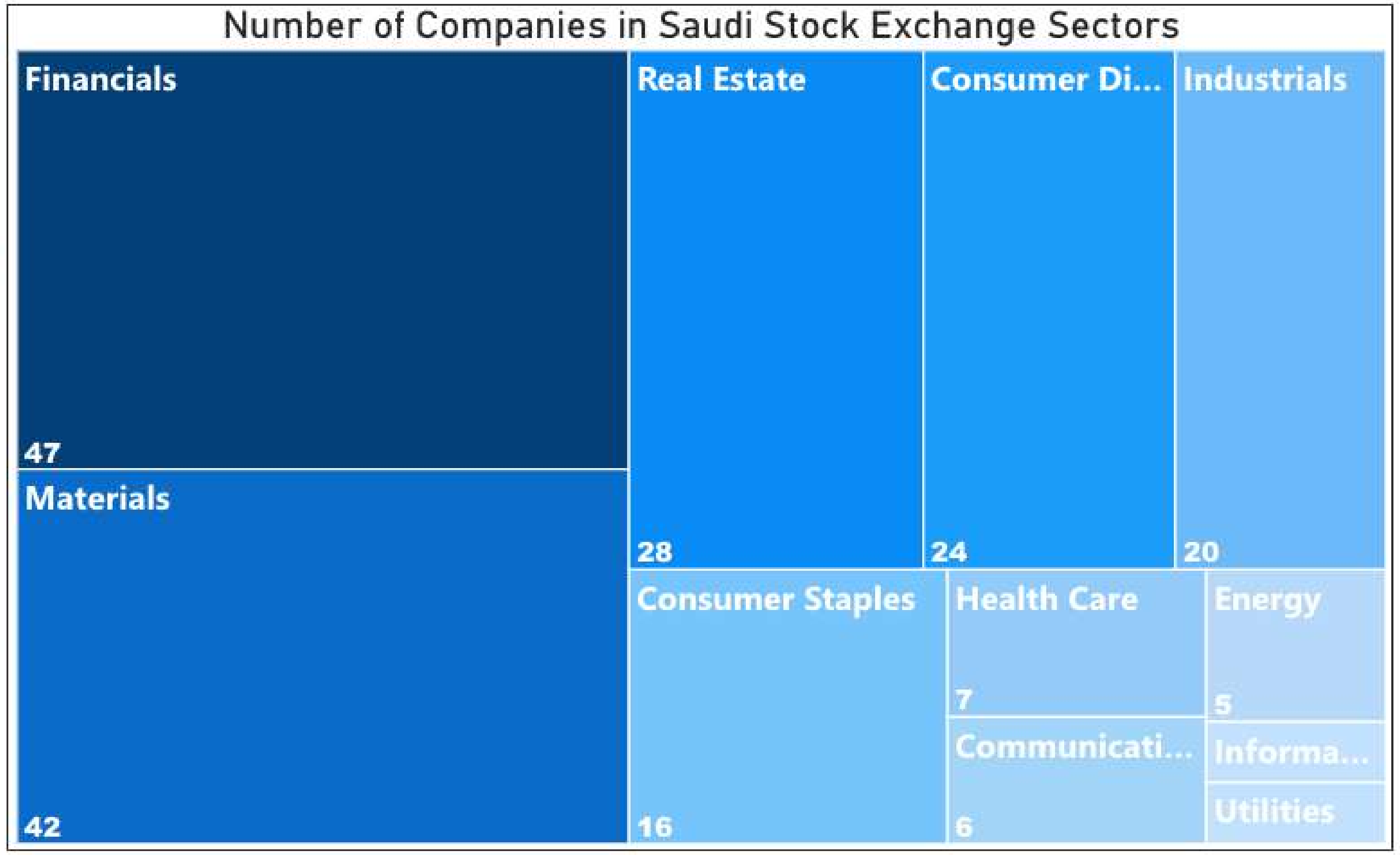
```
#Adding the 'Year' feature by splitting the Date column
df["Year"] = pd.DatetimeIndex(df['Date']).year
df['Month'] = pd.DatetimeIndex(df['Date']).month
df.head()
```

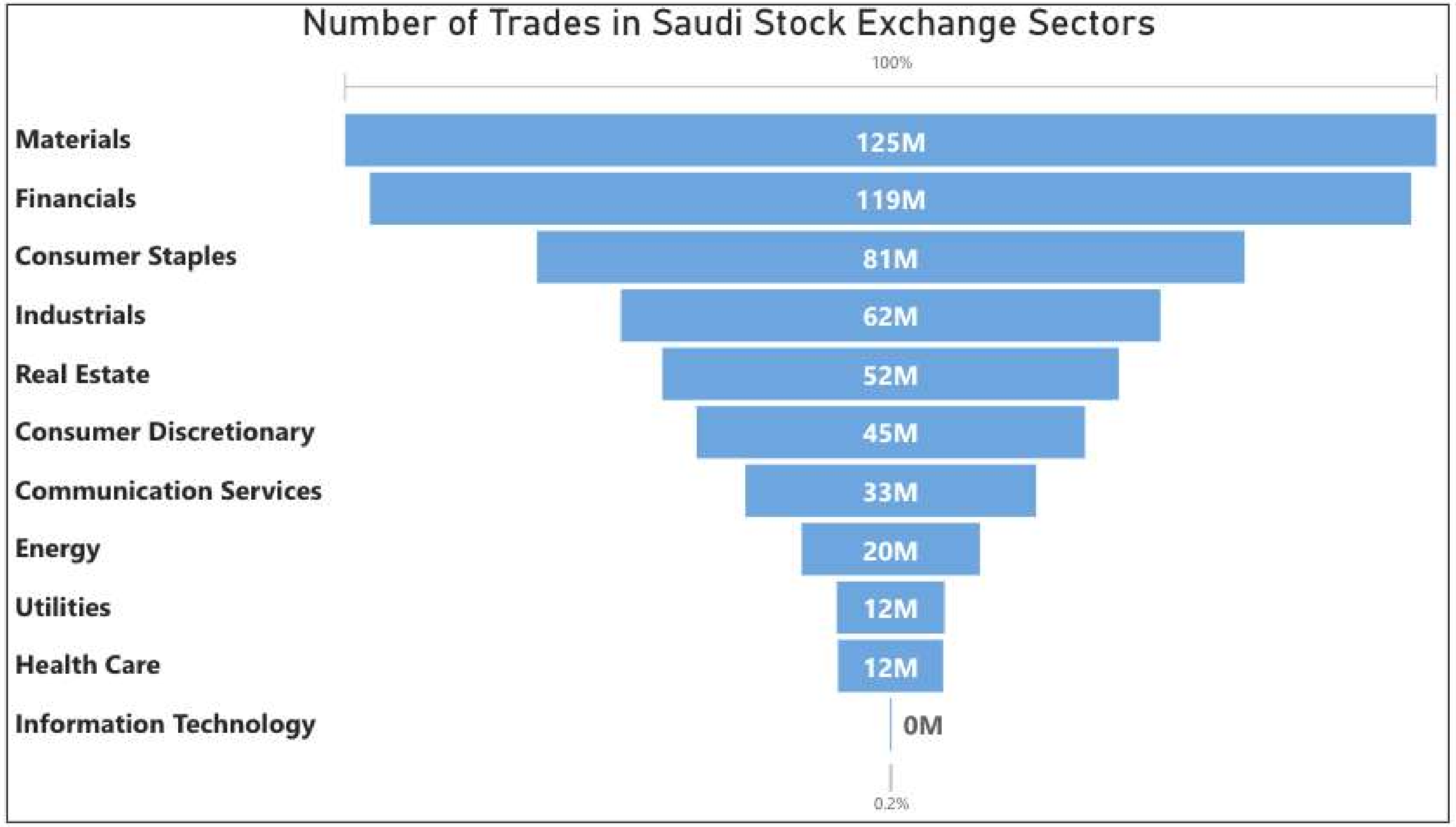
	Company_Name	Sector	Date	Open	High	Low	Close	Price_Change	%_Change	Volume_Traded	Value_Traded	No_of_Trades	Year	Month
0	SARCO	Energy	2020-03-05	35.55	35.85	34.90	34.90	-0.40	-1.13	436609.0	15399073.50	804.0	2020	3
1	SARCO	Energy	2020-03-04	34.70	35.65	34.50	35.30	0.25	0.71	737624.0	25981391.35	1268.0	2020	3
2	SARCO	Energy	2020-03-03	34.70	35.15	34.70	35.05	1.05	3.09	489831.0	17116413.40	854.0	2020	3
3	SARCO	Energy	2020-03-02	35.20	35.65	34.00	34.00	-0.55	-1.59	736157.0	25858700.60	1242.0	2020	3
4	SARCO	Energy	2020-03-01	35.35	35.60	34.25	34.55	-2.05	-5.60	738685.0	25747967.55	1625.0	2020	3

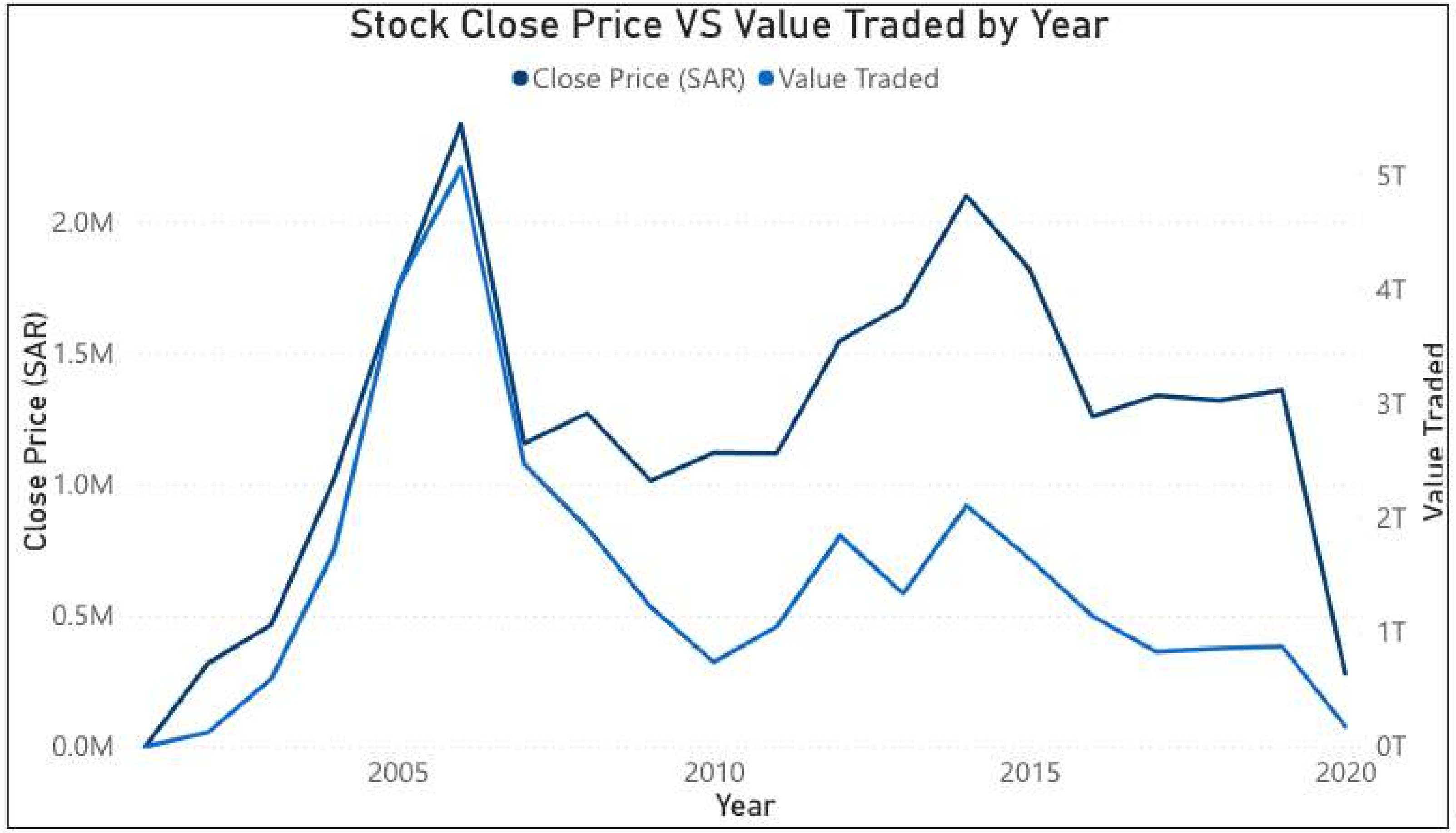
3. Saudi Stock Exchange Dashboard

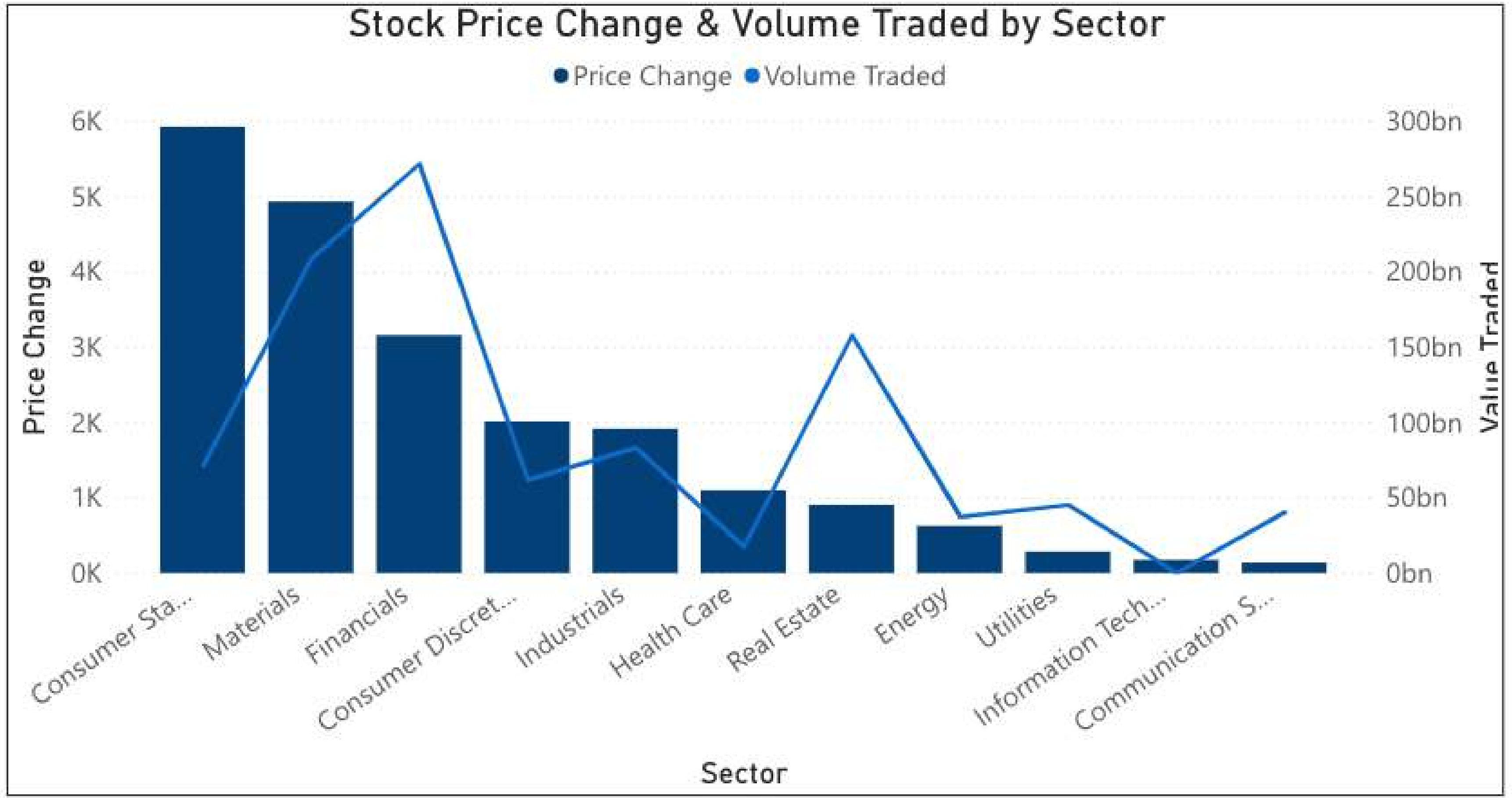
Saudi Stock Exchange Dashboard

Saudi Tadawul Group



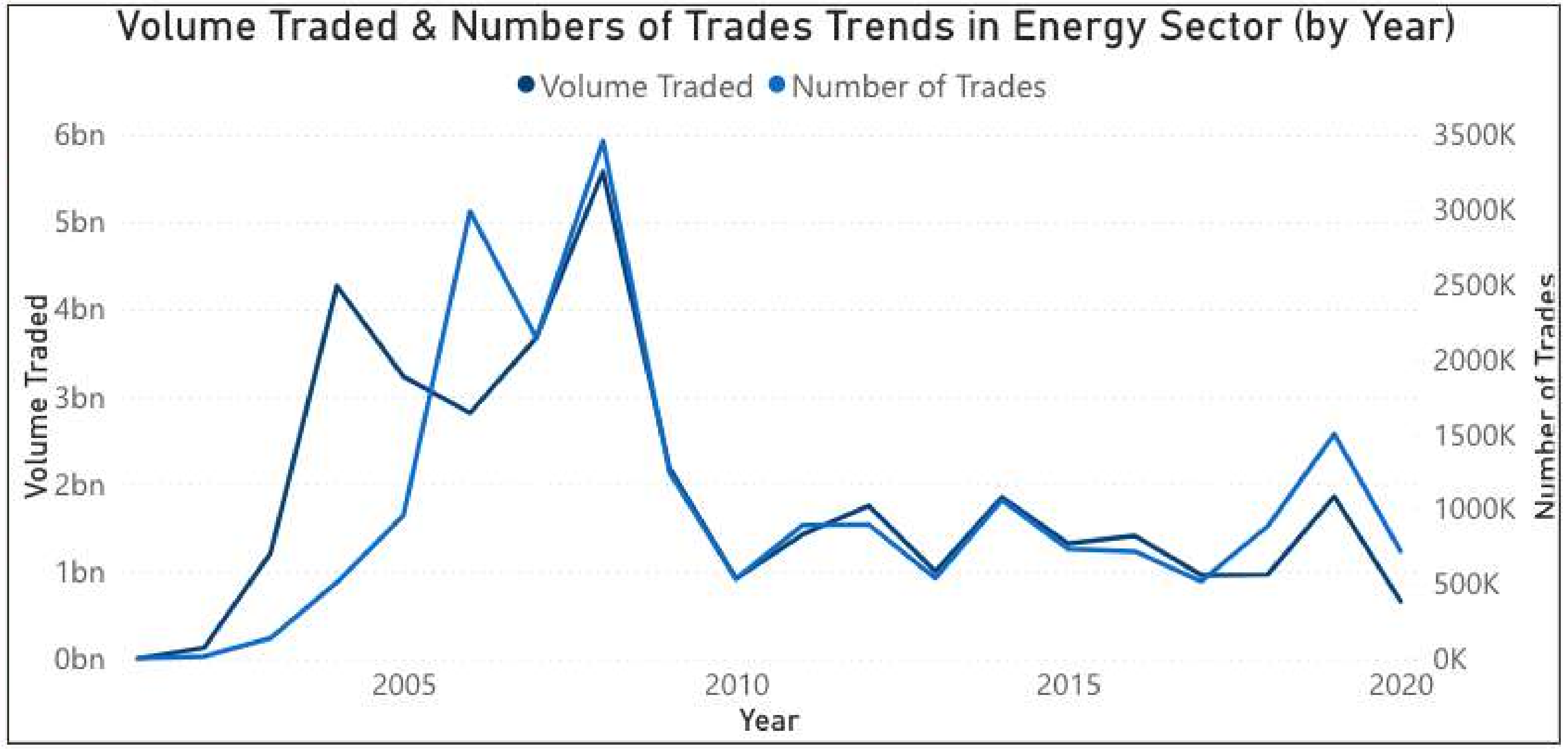


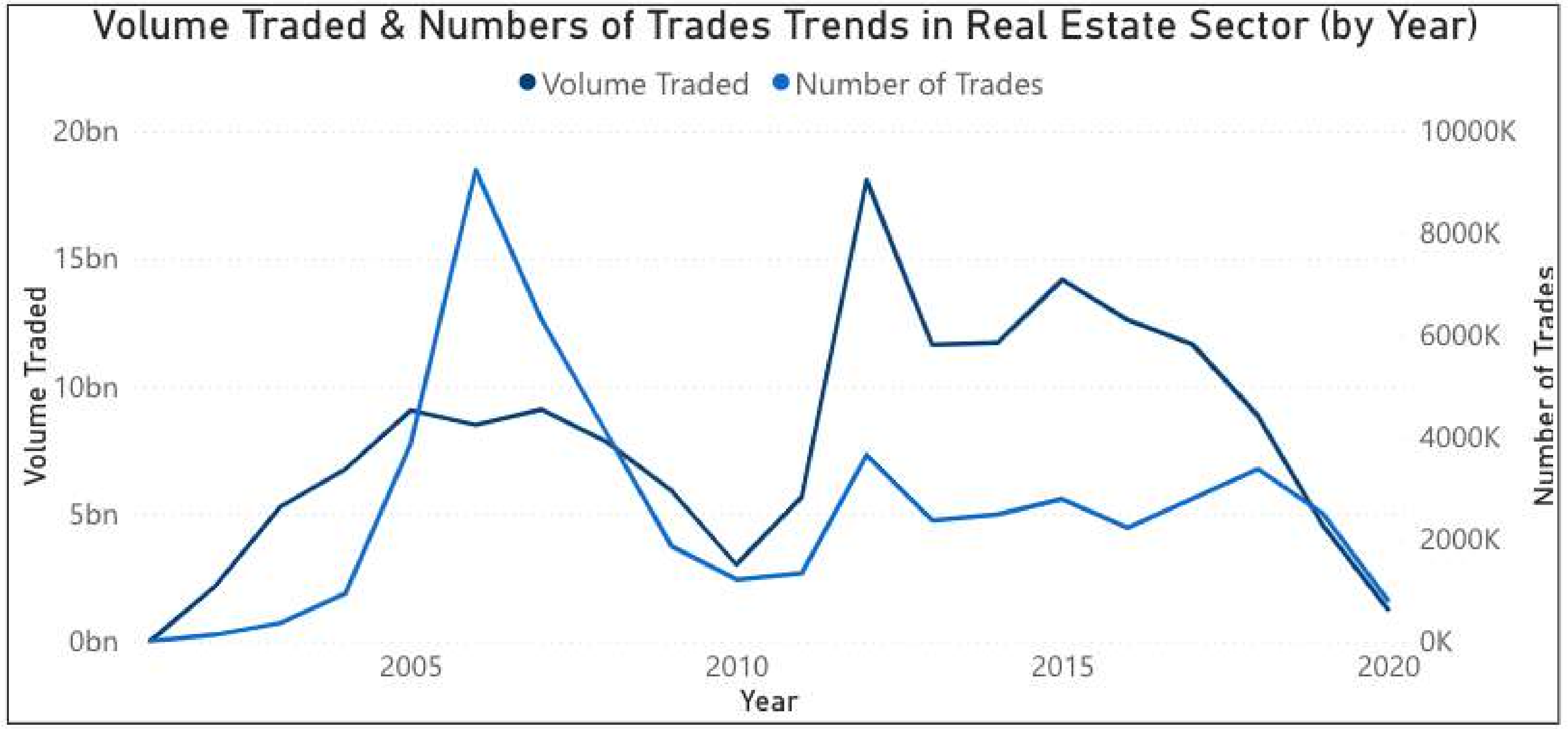


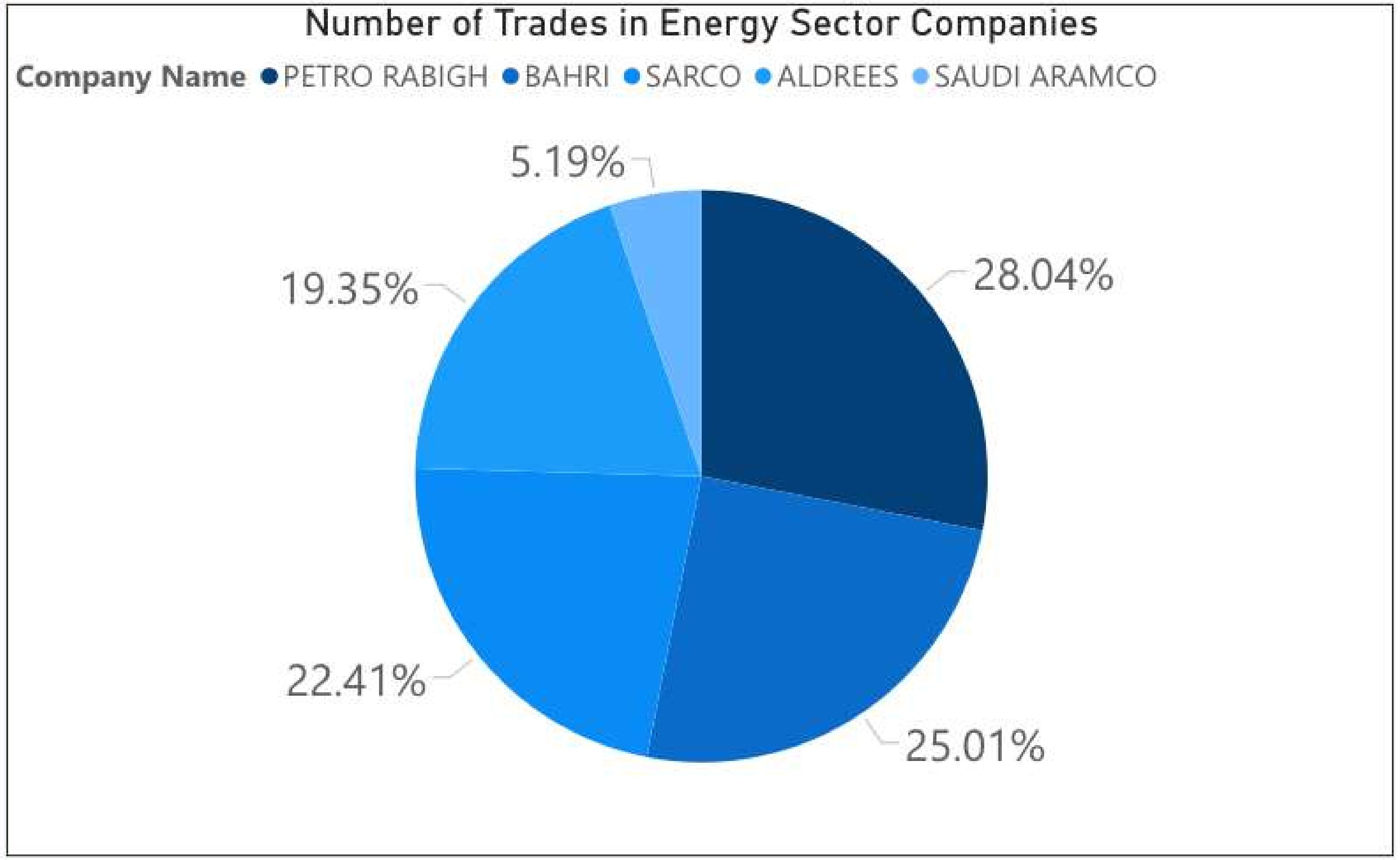


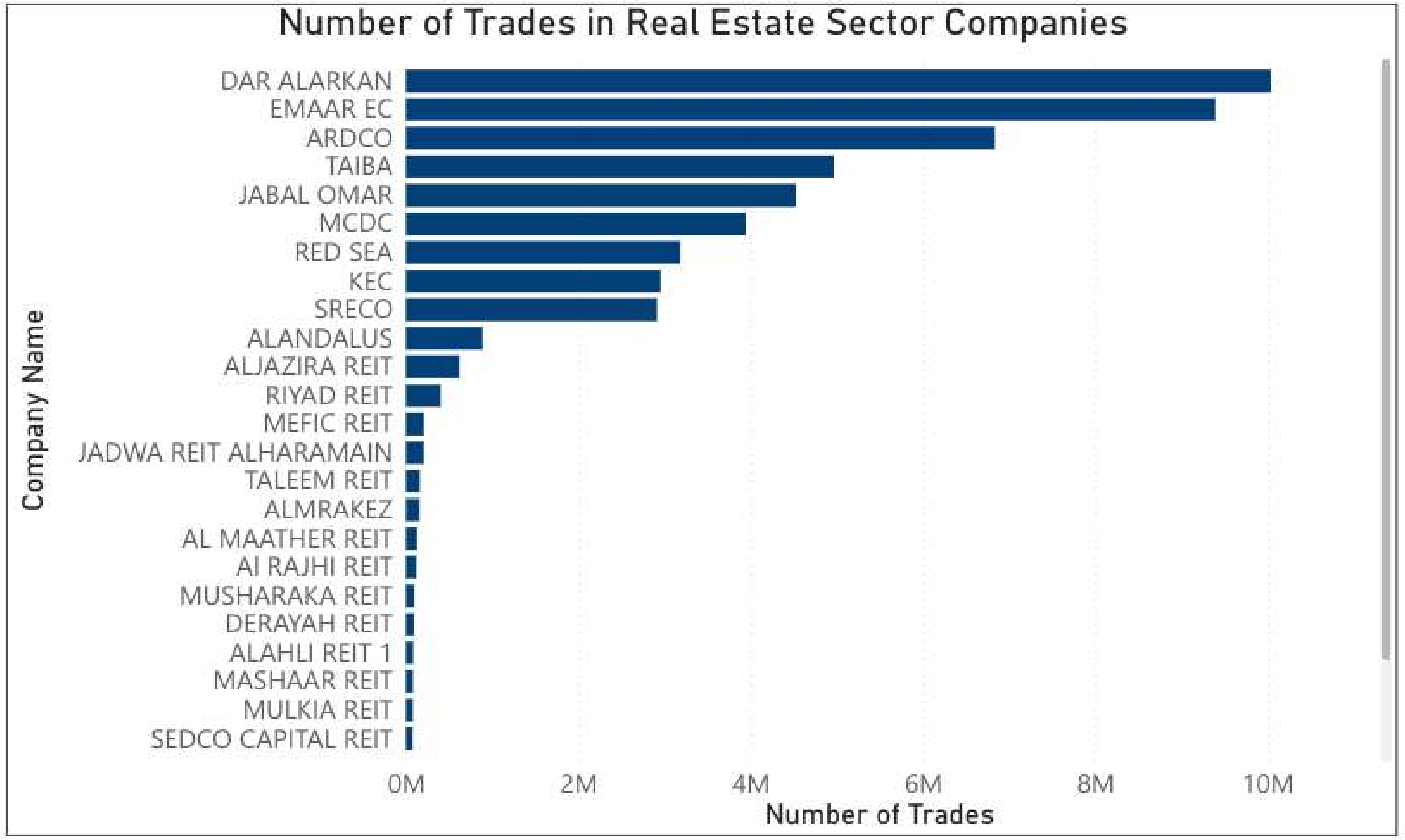
Energy & Real Estate Saudi Stock Exchange Dashboard

مجموعة تداول السعودية
Saudi Tadawul Group

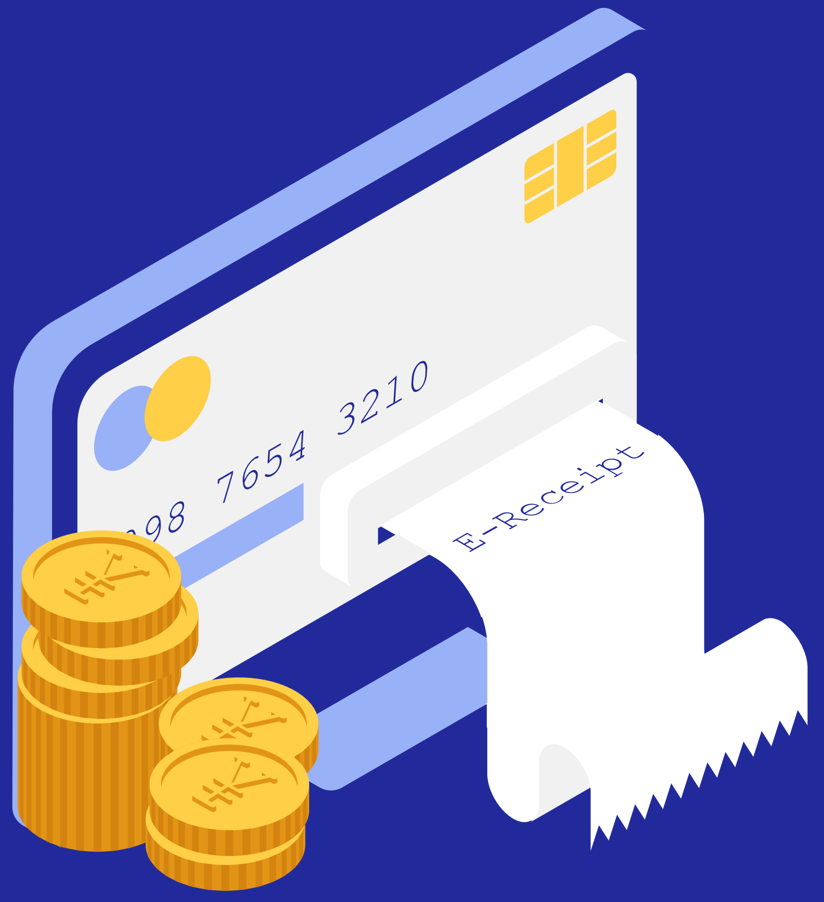






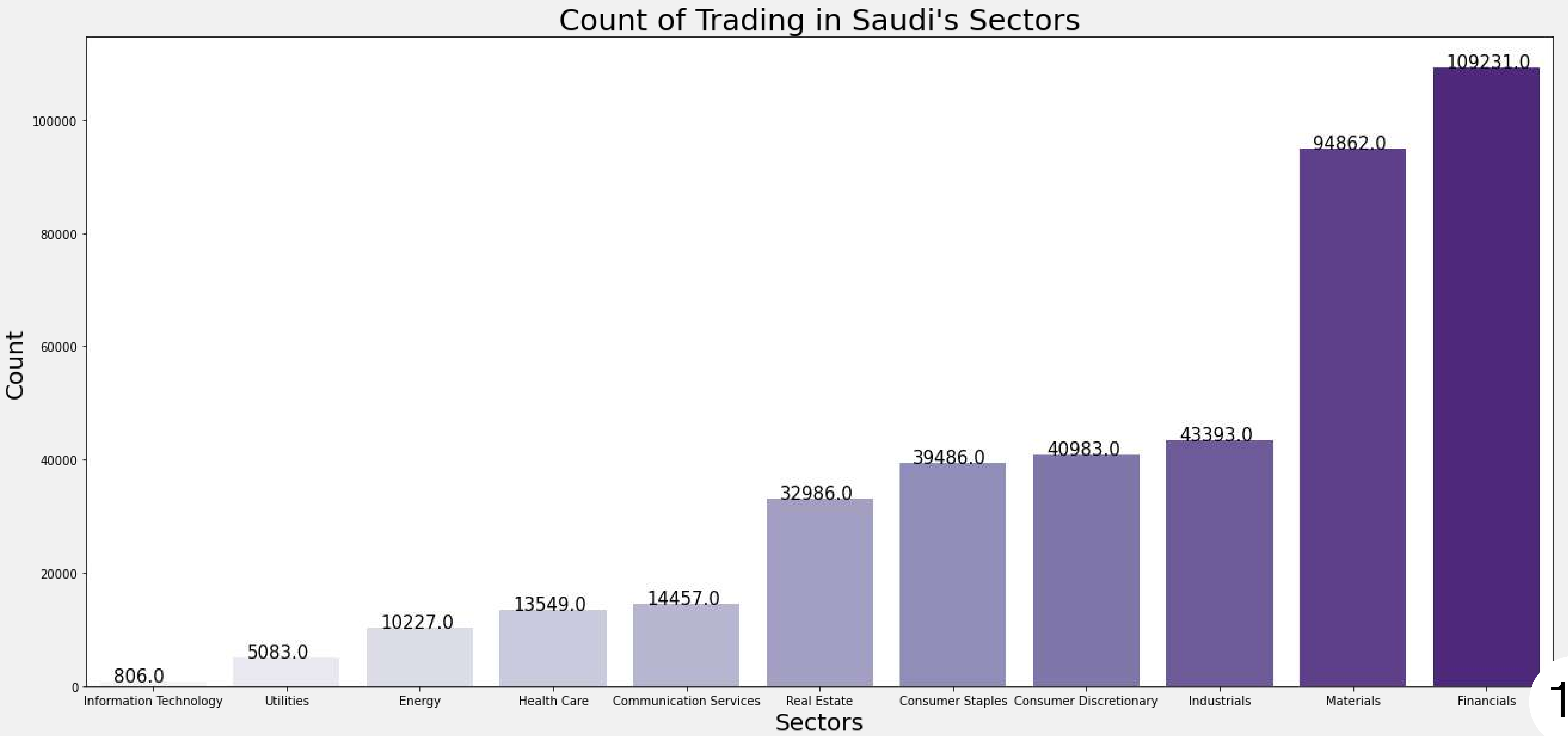


4. Data visualization



Data Visualization

1. Count of Trading in Saudi's Sectors



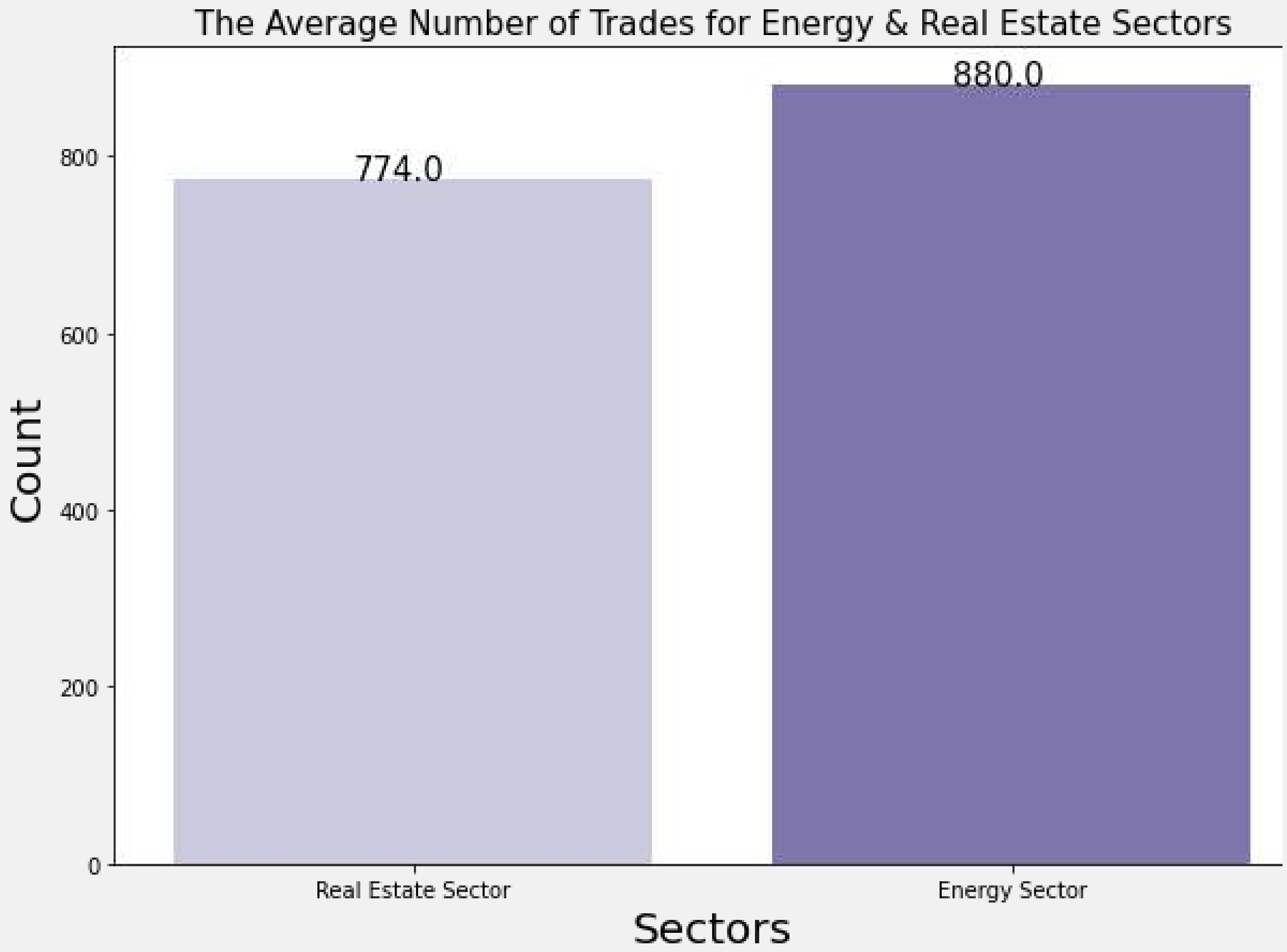
Data Visualization

1. Count of Trading in Saudi's Sectors

```
# Plotting the counter of Trading in Saudi's Sectors

# creating data on which bar chart will be plot
x = df.Sector
y=df.Sector.value_counts()

plt.subplots(figsize=(22,10))


# parameters size
plt.rcParams.update({'font.size': 10})
ax=sns.countplot(df.Sector, palette='Purples',order = df['Sector'].value_counts().index [::-1])

    # sort the sctors values
for p in ax.patches:
    ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.10, p.get_height()+0.1), size=15)

plt.title(" Count of Trading in Saudi's Sectors ",fontsize=25)
plt.xlabel('Sectors' , fontsize=20)
plt.ylabel('Count',fontsize=20)

plt.show()
```

Data Visualization

2. The Average Number of Trades for Energy & Real Estate Sectors



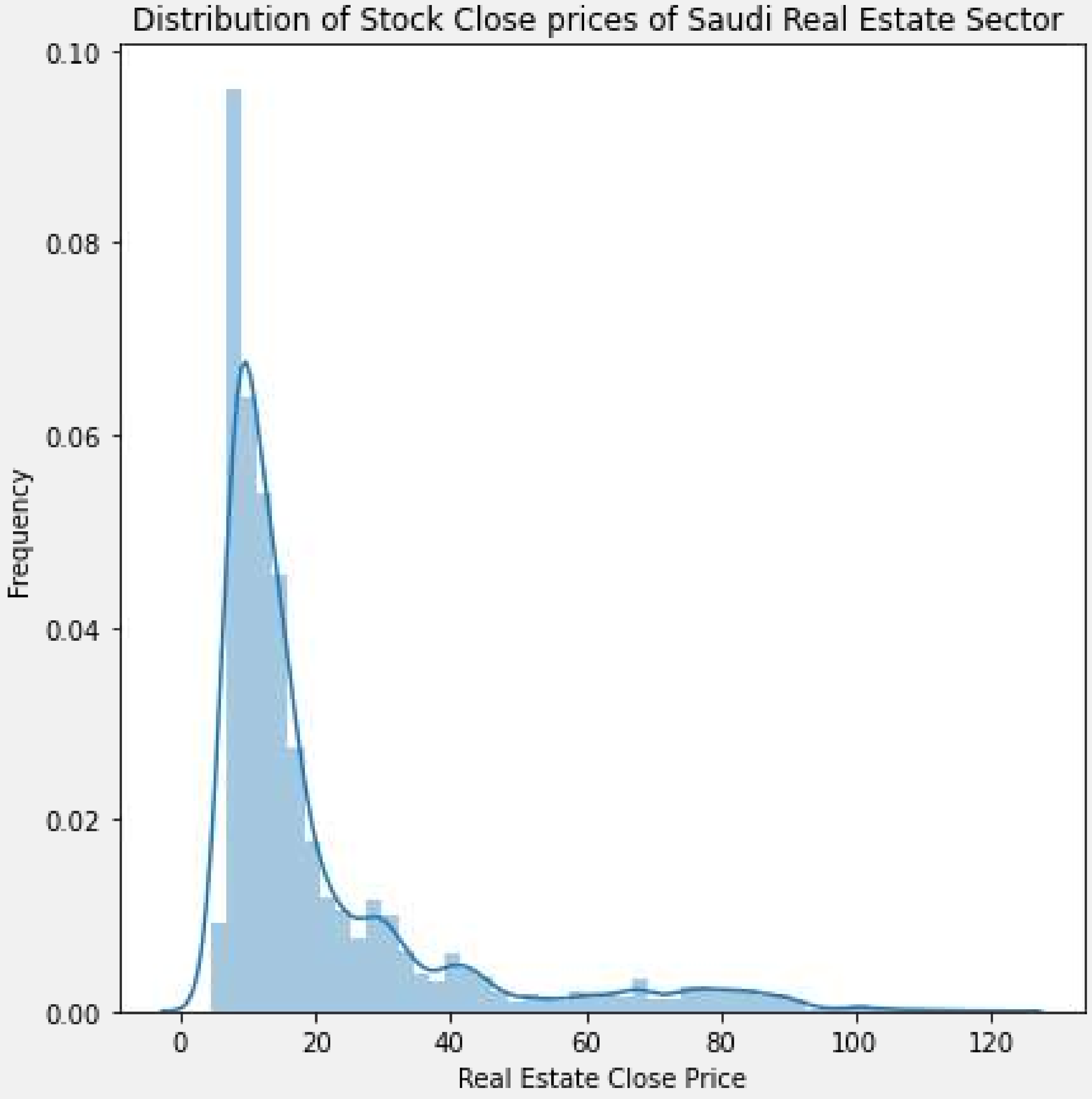
Data Visualization

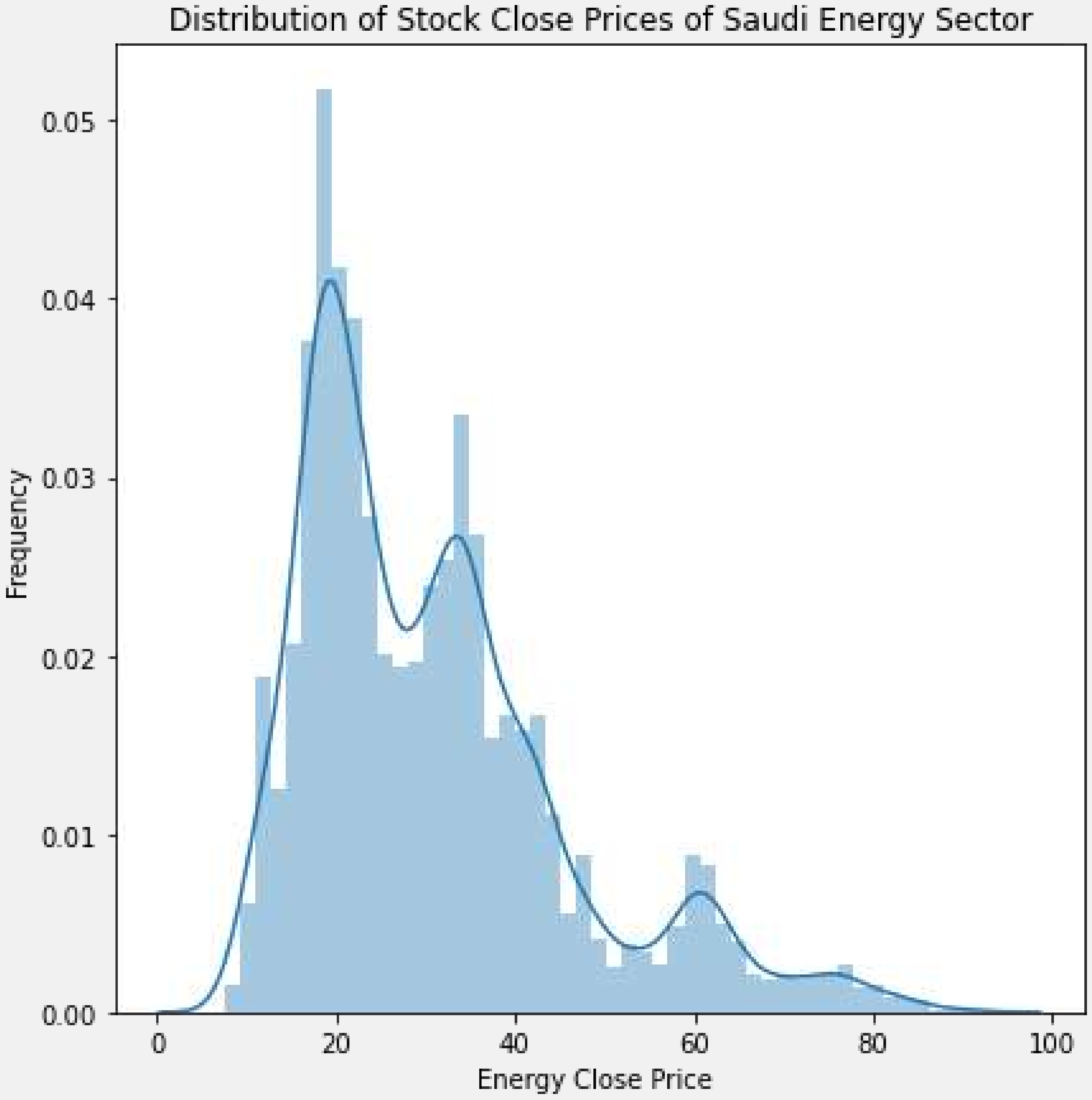
2. The Average Number of Trades for Energy & Real Estate Sectors

```
# Plotting the counter of Trading in Saudi's Sectors

# creating data on which bar chart will be plot
x = ['Real Estate Sector' , 'Energy Sector']
y= [774,880]

plt.subplots(figsize=(10,7))


# parameters size
plt.rcParams.update({'font.size': 10})
ax=sns.barplot(x,y, palette='Purples')

    # sort the sctors values
for p in ax.patches:
    ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.30, p.get_height()+0.1), size=15)

plt.title(" The Average Number of Trades for Energy & Real Estate Sectors ",fontsize=15)
plt.xlabel('Sectors' , fontsize=20)
plt.ylabel('Count',fontsize=20)

plt.show()
```

Data Visualization

3. Distribution of Stock Close Prices of Saudi Energy Sector





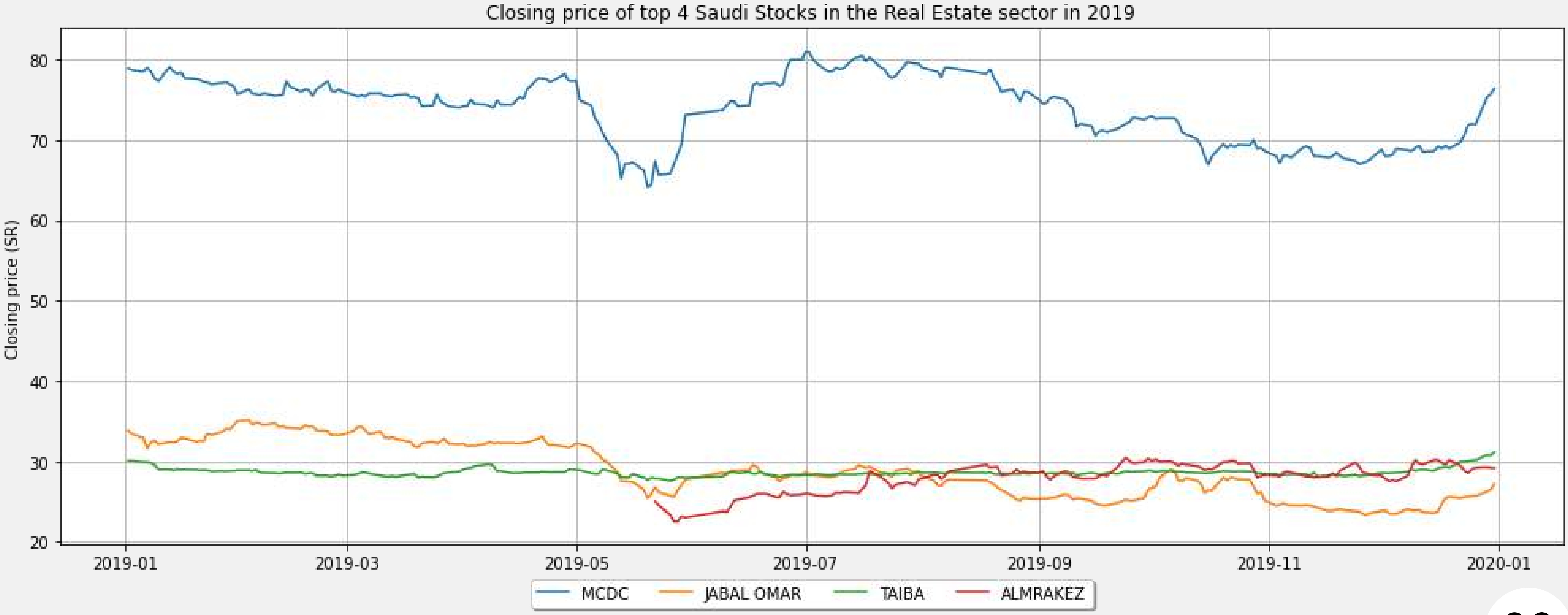
Data Visualization

3. Distribution of Stock Close Prices of Saudi Energy Sector

```
plt.figure(figsize=(15,7))


plt.subplot(1, 2, 1)
sns.distplot(RealEstate_df['Close'])
plt.title('Distribution of Stock Close prices of Saudi Real Estate Sector ')
plt.ylabel('Frequency')
plt.xlabel('Real Estate Close Price')
#---------------------------#

plt.subplot(1, 2, 2)
sns.distplot(Energy_df['Close'])
plt.title('Distribution of Stock Close Prices of Saudi Energy Sector')
plt.ylabel('Frequency')
plt.xlabel('Energy Close Price ')
plt.show()
```

Data Visualization

4. Closing price of top 4 Saudi Stocks in the Real Estate sector in 2019



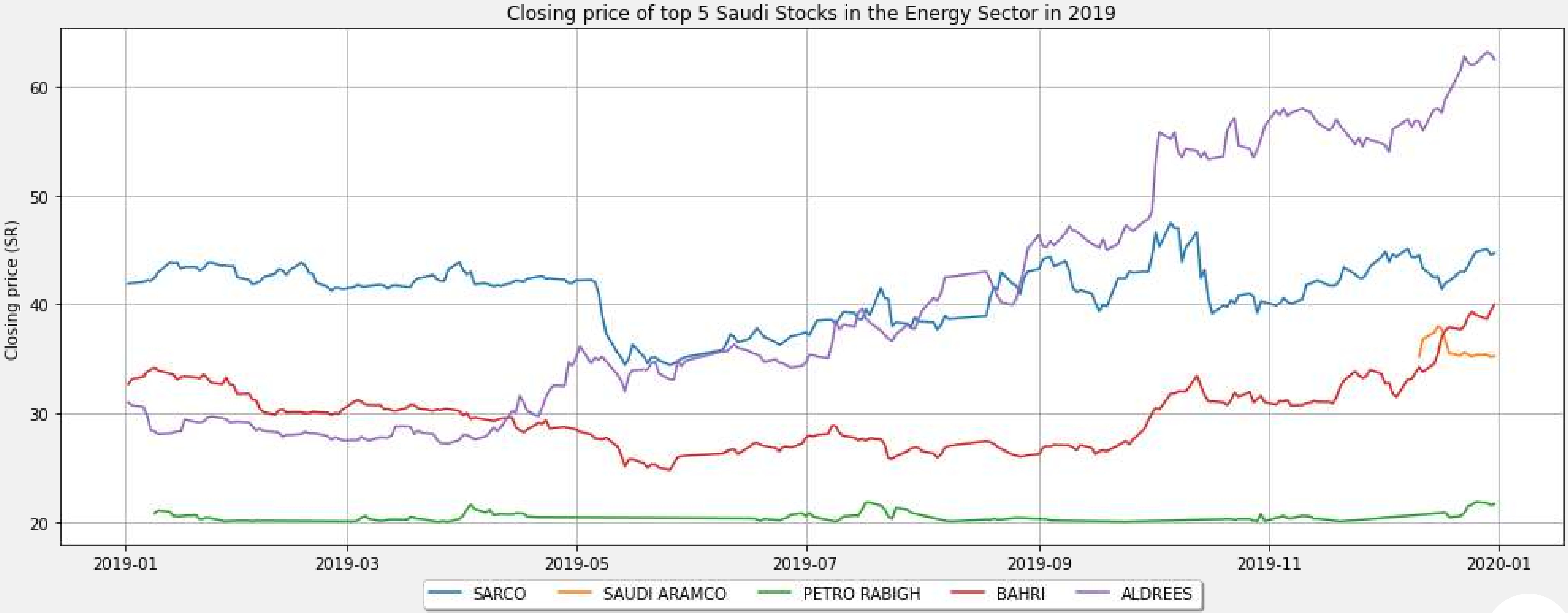
Data Visualization

4. Closing price of top 4 Saudi Stocks in the Real Estate sector in 2019

```
# Visulization of Closing price of Saudi Stocks in the Real Estate sector in 2019

# Take the Financials of 2019 only
RealEstate_2019 = RealEstate_df.loc[(RealEstate_df.Date > '2019-01-01')& (RealEstate_df.Date < '2020-01-01') & (Real

# Set the size of chart as 17 * 6
plt.figure(figsize=(17, 6))

# Line chart
sns.lineplot(x = RealEstate_2019.Date, y="Close", hue = "Company_Name", markers = True, data = RealEstate_2019)
plt.title('Closing price of top 4 Saudi Stocks in the Real Estate sector in 2019')
plt.xlabel('Year')
plt.ylabel('Closing price (SR)')
plt.legend(loc ='upper center', bbox_to_anchor = (0.5, -0.05), fancybox=True, shadow=True, ncol=5)
plt.grid(True)
plt.show()
```

Data Visualization

5. Closing price of top 5 Saudi Stocks in the Energy Sector in 2019



Data Visualization

5. Closing price of top 5 Saudi Stocks in the Energy Sector in 2019

```
# Visulization of Closing price of Saudi Stocks in the Real Estate sector in 2019

# Take the Financials of 2019 only
Energy_df_2019 = Energy_df.loc[(Energy_df.Date > '2019-01-01')& (Energy_df.Date < '2020-01-01') & (Energy_df.Close >

# Set the size of chart as 17 * 6
plt.figure(figsize=(17, 6))

# Line chart
sns.lineplot(x = Energy_df.Date, y="Close", hue = "Company_Name", markers = True, data = Energy_df_2019)
plt.title('Closing price of top 5 Saudi Stocks in the Energy Sector in 2019')
plt.xlabel('Year')
plt.ylabel('Closing price (SR)')
plt.legend(loc ='upper center', bbox_to_anchor = (0.5, -0.05), fancybox=True, shadow=True, ncol=5)
plt.grid(True)
plt.show()
```

5. Machine Learning Models

Machine Learning Models

Real Estate Sector

Real Estate head

Display the first five rows of Real Estate sector

In [80]:
```
# Disply first five rows of data
RealEstate_df.head()
```

Out[80]:

	Company_Name	Sector	Date	Open	High	Low	Close	Price_Change	%_Change	Volume_Traded	Value_Traded	No_of_Trades	Year	Month
549156	RIYAD REIT	Real Estate	2020-03-05	8.69	8.96	8.69	8.91	0.30	3.48	329074.0	2919780.78	405.0	2020	3
549157	RIYAD REIT	Real Estate	2020-03-04	8.69	8.75	8.56	8.61	0.03	0.35	649286.0	5628851.72	320.0	2020	3
549158	RIYAD REIT	Real Estate	2020-03-03	8.48	8.64	8.48	8.58	0.14	1.66	242168.0	2079110.00	241.0	2020	3
549159	RIYAD REIT	Real Estate	2020-03-02	8.30	8.57	8.30	8.44	0.14	1.69	132201.0	1116460.13	217.0	2020	3
549160	RIYAD REIT	Real Estate	2020-03-01	8.48	8.48	8.30	8.30	-0.20	-2.35	174369.0	1466967.87	365.0	2020	3

Prepare and Split the Data

Create x, y data, label Encoding the company name column, and split the data

In [142]:
```
# Create my X, y data

target = 'No_of_Trades' # Target Varible
features = ['Company_Name','Value_Traded','Volume_Traded','Price_Change','%_Change','Year','Month','Open','Close']


X = RealEstate_df[features]
y = RealEstate_df[target]
```

In [143]:
```
# Label Encoding the "Company Name" column

le = LabelEncoder()

X.iloc[:,0] = le.fit_transform(X.iloc[:,0])
```

In [144]:
```
# Train Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, random_state=2)
```

Linear Regression Model

Create a Linear Regression model and calculate the score

```
In [41]:  # Create a model object
          lr = LinearRegression()

          # Train the model
          lr.fit(X_train, y_train)

Out[41]:  LinearRegression()

In [42]:  # Score the Model using cross validation

          crossVal=cross_val_score(
              lr, #model
              X_train,
              y_train,
              cv=5,
              scoring="neg_mean_absolute_error" # scoring metric to use
          ).mean()
          print("The cross value is %.2f" % crossVal)

          The cross value is -314.23

In [43]:  # Predict on Test Data
          preds = lr.predict(X_test)

In [112]: # Calculate The Linear Regression Score
          lrScore=r2_score(y_true=y_test, y_pred=preds)

          # get the score percent
          lrPercent=(lrScore)*100

          # display result with 2 digits
          lrScore=float('{0:.2f}'.format(lrPercent))

          print('The Linear Regression Score ', lrScore)

          The Linear Regression Score  81.9
```

Decision Tree Regression Model

Create a Decision Tree model and calculate the score

In [119]:
```
# Create a model object
reg_tree = DecisionTreeRegressor(random_state = 0, max_depth= 4, criterion= 'mse')

# Fit the model
reg_tree.fit(X_train, y_train)

# Predict on Test Data
preds_tree = reg_tree.predict(X_test)
```

In [120]:
```
# Calculate The Decision tree Score
RegTreeScore=r2_score(y_true=y_test, y_pred=preds_tree)

# get the score percent
RegTreePercent=(RegTreeScore)*100

# display result with 2 digits
RegTreeScore=float('{0:.2f}'.format(RegTreePercent))

print('The Decision tree Score ',RegTreeScore)
```

```
The Decision tree Score  80.47
```

Random Forest Regression Model

Create a Random Forest model and calculate the score

In [115]:
```
# Create a model object
reg_forest = RandomForestRegressor(n_estimators = 10, random_state = 0, criterion = 'mse')

# Fit the model
reg_forest.fit(X_train, y_train)

# Predict on Test Data
preds_forest = reg_forest.predict(X_test)
```

In [118]:
```
# Calculate The Random Forest Score
RegForestScore=r2_score(y_true=y_test, y_pred=preds_forest)

RegForestPercent=(RegForestScore)*100

RegForestScore=float('{0:.2f}'.format(RegForestPercent))
print('The Random Forest Score ', RegForestScore)
```

```
The Random Forest Score  93.6
```

Baseline & Models Evaluation

Create a calc_cost function to calculate the MSE, MAE, RMSE

In [56]:

```
# Calculates the cost functions
def calc_cost(y_true, y_predict):

    "Calculate Cost Functions and print output"

    result_dict = {}

    mse = mean_squared_error(y_true, y_predict)
    mae = mean_absolute_error(y_true, y_predict)
    rmse = mean_squared_error(y_true, y_predict, squared=False)

    # Round the numbers for readability --> decimal points are not important
    ls = [round(mse), round(mae), round(rmse)]
    ls2 = ["MSE", "MAE", "RMSE"]

    for x in range(len(ls)):
        print(f"{ls2[x]}: {ls[x]}")
        result_dict[ls2[x]] = ls[x]

    return result_dict

# Save results to object and print results
print("Baseline")

# The baseline model --> replace values by the mean and calculate the cost functions
# Baseline is concerned with the average value
b_preds = [y_test.mean() for x in range(len(y_test))]

res0 = calc_cost(y_test, b_preds)

print("\nLinear Regression")
res1 = calc_cost(y_test, preds)
print("\nDecision Tree  Regression")
res2= calc_cost(y_test, preds_tree)
print("\nRandom Forest Tree  Regression")
res3=calc_cost(y_test, preds_forest)
```

```
Baseline
MSE:  2892826
MAE:  826
RMSE:  1701

Linear Regression
MSE:  523710
MAE:  312
RMSE:  724

Decision Tree  Regression
MSE:  565009
MAE:  275
RMSE:  752

Random Forest Tree  Regression
MSE:  185249
MAE:  146
RMSE:  430
```

Baseline VS Prediction Models

Comparison between the Baseline and our Prediction models

In [96]:

```
# Comparing baseline vs. our prediction models
print("\nBaseline VS Linear Regression")
b1=res0['MSE']-res1['MSE']
print(b1)

print("\nBaseline VS Decision Tree Regression")
b2=res0['MSE']-res2['MSE']
print(b2)

print("\nBaseline VS Random Forest Regression")
b3=res0['MSE']-res3['MSE']
print(b3)
```

```
Baseline VS Linear Regression
2369116

Baseline VS Decision Tree Regression
2327817

Baseline VS Random Forest Regression
2707577
```

Since the value is positive --> the regression models is better than the Baseline model

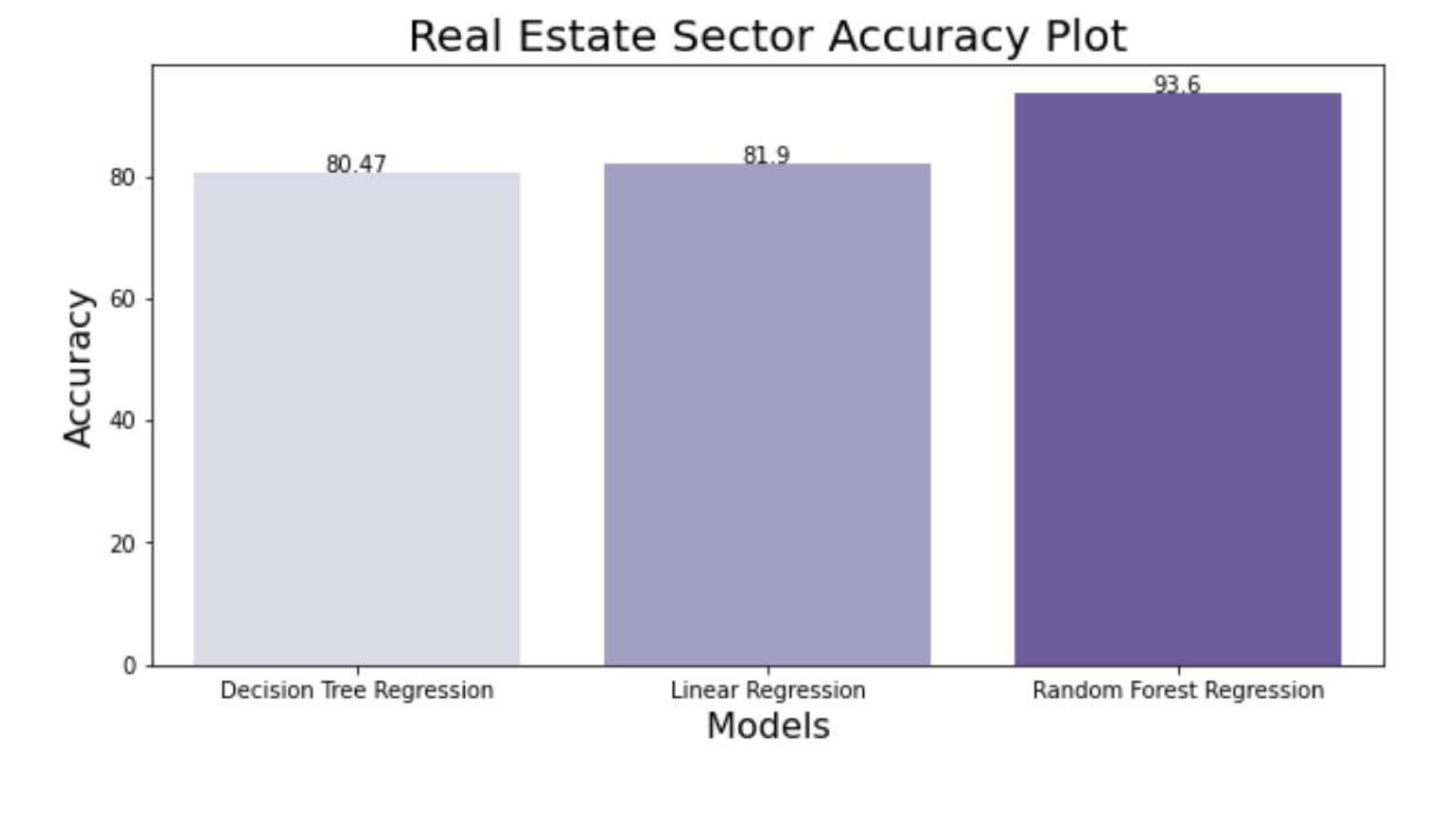
Sector Accuracy Plot

Real Estate Sector Accuracy Plot

```python
# function to add value labels
def sector_accuracy(x,y):
    for i in range(len(x)):
        plt.text(i, y[i], y[i], ha = 'center')

if __name__ == '__main__':

    # creating data on which bar chart will be plot
    x = ['Decision Tree Regression','Linear Regression','Random Forest Regression']
    y=[RegTreeScore,lrScore,RegForestScore]

    # setting figure size by using figure() function
    plt.figure(figsize = (10, 5))

    # making the bar chart on the data
    sns.barplot(x,y, palette = 'Purples')

    # calling the function to add value labels
    sector_accuracy(x, y)

    # giving title to the plot
    plt.title("Real Estate Sector Accuracy Plot",fontsize=20)

    # giving X and Y labels
    plt.xlabel(" Models ",fontsize=16)
    plt.ylabel("Accuracy", fontsize=16)

    # visualizing the plot
    plt.show()
```

Sector Accuracy Plot

Real Estate Sector Accuracy Plot



Machine Learning Models

Energy Sector

Energy head

Display the first five rows of Energy sector

```
In [84]: Energy_df.head()
```

Out[84]:

	Company_Name	Sector	Date	Open	High	Low	Close	Price_Change	%_Change	Volume_Traded	Value_Traded	No_of_Trades	Year	Month
14843	ALDREES	Energy	2010-01-02	15.88	15.88	15.71	15.77	0.0	0.00	79578.0	1257740.10	90.0	2010	1
2541	SARCO	Energy	2010-01-02	49.80	50.25	49.70	49.90	0.1	0.20	76316.0	3809644.65	200.0	2010	1
10086	BAHRI	Energy	2010-01-02	17.80	18.20	17.65	18.15	0.4	2.25	1885686.0	33710771.35	518.0	2010	1
7061	PETRO RABIGH	Energy	2010-01-02	35.80	35.90	35.60	35.70	0.2	0.56	864899.0	30892887.30	680.0	2010	1
7060	PETRO RABIGH	Energy	2010-01-03	35.80	36.00	35.50	35.80	0.1	0.28	1013744.0	36204432.70	664.0	2010	1

Prepare and Split the Data

Create x, y data, label Encoding the company name column, and split the data

In [158]:

```
# Create my X, y data

target = 'No_of_Trades' # Target Varible
features = ['Company_Name','Value_Traded','Volume_Traded','Price_Change','%_Change','Year','Month','Open','Close']


X_E = Energy_df[features]
y_E = Energy_df[target]
```

In [159]:

```
# Label Encoding the "Company Name" column

le_E = LabelEncoder()

X_E.iloc[:,0] = le.fit_transform(X_E.iloc[:,0])
```

In [160]:

```
# Train Test Split
X_train_E, X_test_E, y_train_E, y_test_E = train_test_split(X_E, y_E, train_size=0.75, random_state=2)
```

Linear Regression Model

Create a Linear Regression model and calculate the score

```
In [58]:  # Create a model object
          lr_E = LinearRegression()

          # Fit the model
          lr_E.fit(X_train_E, y_train_E)

Out[58]:  LinearRegression()
```

```
In [59]:  # Score the Model using cross validation

          crossVal=cross_val_score(
              lr_E, #model
              X_train_E,
              y_train_E,
              cv=5,
              scoring="neg_mean_absolute_error" # scoring metric to use
          ).mean()
          print("The cross value is %.2f" % crossVal)

          The cross value is -332.66
```

```
In [60]:  # Predict on Test Data
          preds_E = lr_E.predict(X_test_E)
```

```
In [110]: # Calculate The Linear Regression Score
          lrScore_E=r2_score(y_true=y_test_E, y_pred=preds_E)

          # get the score percent
          lrPercent_E=(lrScore_E)*100

          # display result with 2 digits
          lrScore_E=float('{0:.2f}'.format(lrPercent_E))

          print('The Linear Regression Score',lrScore_E)

          The Linear Regression Score 75.96
```

Decision Tree Regression Model

Create a Decision Tree model and calculate the score

```
In [105]:
# Create a model object
reg_tree_E = DecisionTreeRegressor(random_state = 0, max_depth= 4, criterion= 'mse')

# Fit the model
reg_tree_E.fit(X_train_E, y_train_E)

# Predict on Test Data
preds_tree_E = reg_tree_E.predict(X_test_E)
```

```
In [107]:
# Calculate The Decision tree Score
RegTreeScore_E=r2_score(y_true=y_test_E, y_pred=preds_tree_E)

#get the score percent
RegTreePercent_E=(RegTreeScore_E)*100

# display result with 2 digits
RegTreeScore_E=float('{0:.2f}'.format(RegTreePercent_E))

print('The Decision tree Score',RegTreeScore_E)
```

```
The Decision tree Score 88.66
```

Random Forest Regression Model

Create a Random Forest model and calculate the score

```
In [98]:  # Create a model object
          reg_forest_E = RandomForestRegressor(n_estimators = 10, random_state = 0, criterion = 'mse')

          # Fit the model
          reg_forest_E.fit(X_train_E, y_train_E)

          # Predict on Test Data
          preds_forest_E = reg_forest_E.predict(X_test_E)
```

```
In [104]: # Calculate The Random Forest Score
          RegForestScore_E=r2_score(y_true=y_test_E, y_pred=preds_forest_E)
              #get the score percent
          RegForestPercent_E=(RegForestScore_E)*100
              # display result with 2 digits

          RegForestScore_E=float('{0:.2f}'.format(RegForestPercent_E))
          print('The Random Forest Score',RegForestScore_E)

          The Random Forest Score 94.74
```

Baseline & Models Evaluation

Create a calc_cost function to calculate the MSE, MAE, RMSE

In [72]:

```
# Calculates the cost functions
def calc_cost(y_true, y_predict):

    "Calculate Cost Functions and print output"

    result_dict = {}

    mse = mean_squared_error(y_true, y_predict)
    mae = mean_absolute_error(y_true, y_predict)
    rmse = mean_squared_error(y_true, y_predict, squared=False)

    # Round the numbers for readability --> decimal points are not important
    ls = [round(mse), round(mae), round(rmse)]
    ls2 = ["MSE", "MAE", "RMSE"]

    for x in range(len(ls)):
        print(f"{ls2[x]}: {ls[x]}")
        result_dict[ls2[x]] = ls[x]

    return result_dict

# Save results to object and print results
print("Baseline")

# The baseline model --> replace values by the mean and calculate the cost functions
# Baseline is concerned with the average value
b_preds = [y_test_E.mean() for x in range(len(y_test_E))]

res0 = calc_cost(y_test_E, b_preds)

print("\nLinear Regression")
res1 = calc_cost(y_test_E, preds_E)
print("\nDecision Tree  Regression")
res2= calc_cost(y_test_E,  preds_tree_E)
print("\nRandom Forest Tree  Regression")
res3=calc_cost(y_test_E,  preds_forest_E)
```

```
Baseline
MSE:  3473576
MAE:  688
RMSE:  1864

Linear Regression
MSE:  835074
MAE:  339
RMSE:  914

Decision Tree  Regression
MSE:  393990
MAE:  315
RMSE:  628

Random Forest Tree  Regression
MSE:  182848
MAE:  153
RMSE:  428
```

Baseline VS Prediction Models

Comparison between the Baseline and our Prediction models

In [173]:

```
print("\nBaseline VS Linear Regression")
b1=res0['MSE']-res1['MSE']
print(b1)

print("\nBaseline VS Decision Tree Regression")
b2=res0['MSE']-res2['MSE']
print(b2)

print("\nBaseline VS Random Forest Regression")
b3=res0['MSE']-res3['MSE']
print(b3)
```

```
Baseline VS Linear Regression
2638502

Baseline VS Decision Tree Regression
3079586

Baseline VS Random Forest Regression
3290728
```

Since the value is positive --> the regression models is better than the Baseline model

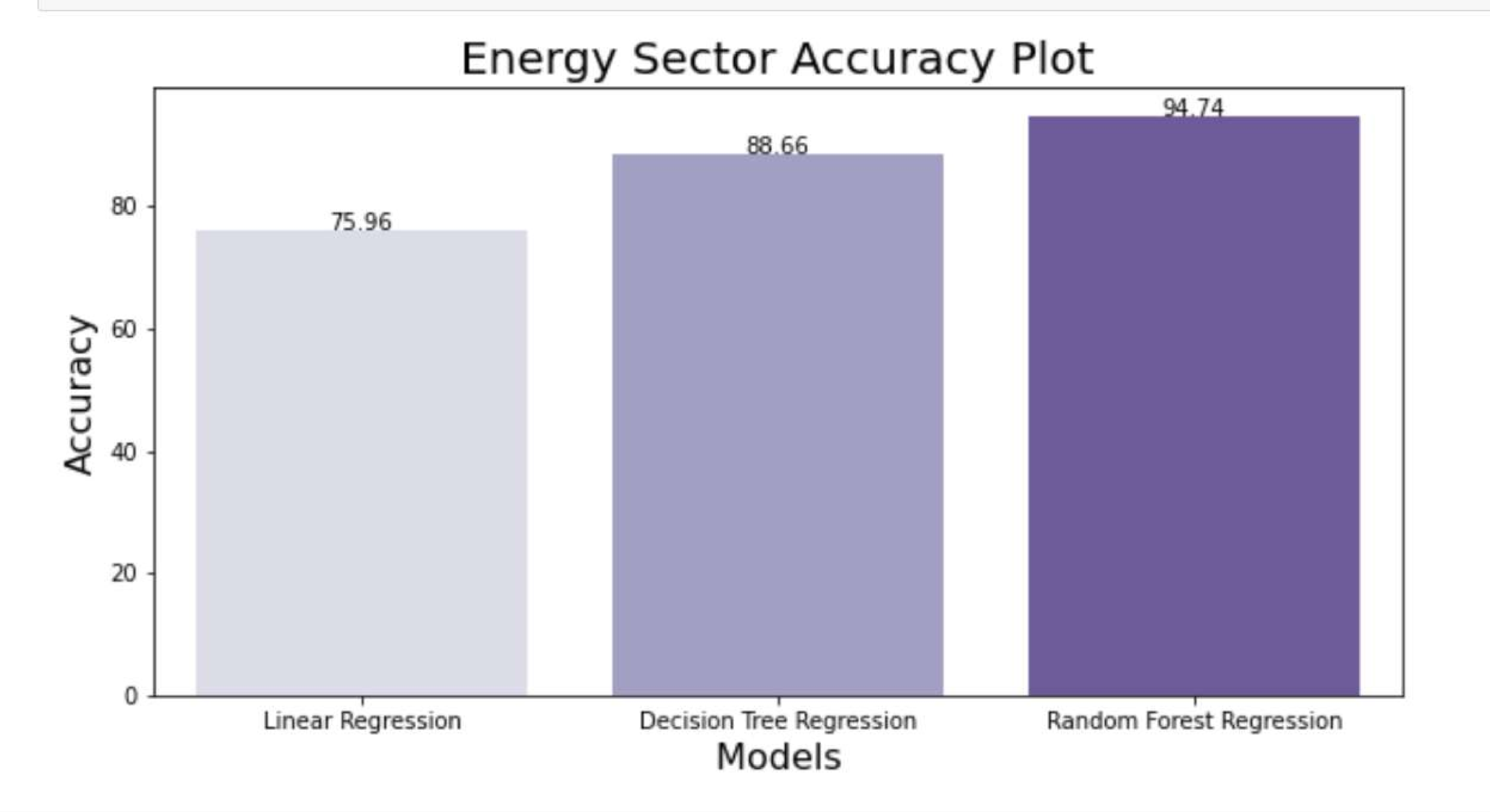
Sector Accuracy Plot

Energy Sector Accuracy Plot

```python
# function to add value labels
def sector_accuracy(x,y):
    for i in range(len(x)):
        plt.text(i, y[i], y[i], ha = 'center')

if __name__ == '__main__':

    # creating data on which bar chart will be plot
    x = ['Linear Regression','Decision Tree Regression','Random Forest Regression']
    y=[lrScore_E,RegTreeScore_E,RegForestScore_E]

    # setting figure size by using figure() function
    plt.figure(figsize = (10, 5))

    # making the bar chart on the data
    sns.barplot(x,y, palette = 'Purples')

    # calling the function to add value labels
    sector_accuracy(x, y)

    # giving title to the plot
    plt.title("Energy Sector Accuracy Plot",fontsize=20)

    # giving X and Y labels
    plt.xlabel(" Models ",fontsize=16)
    plt.ylabel("Accuracy", fontsize=16)

    # visualizing the plot
    plt.show()
```

Sector Accuracy Plot

Energy Sector Accuracy Plot

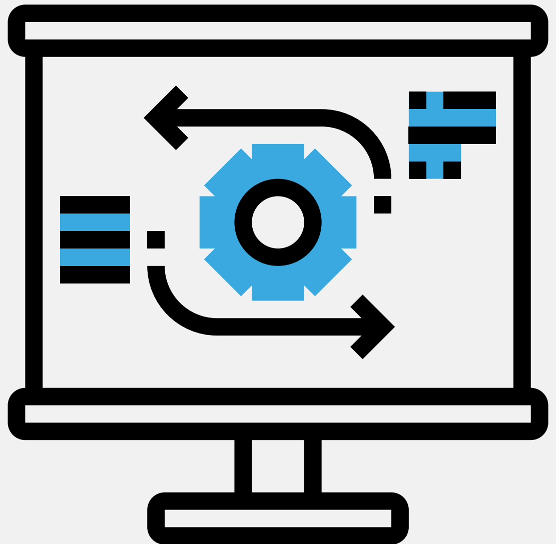


6. Model Optimization - Hyperparameter Tuning

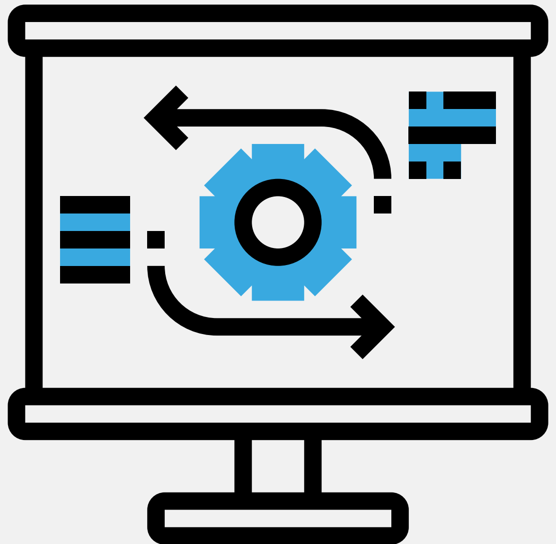
1- Real Estate Sector (Grid Search)

```
param_grid = {
    "n_estimators": (50,100), # how many trees in our forest
    "max_depth": (0,100) # how deep each decision tree can be

}

grid = GridSearchCV(
    reg_forest,
    param_grid,
    cv = 5,
    n_jobs=-1,
    verbose=1
)

grid.fit(X_train, y_train)
```

```
# Model optimization evaluation
print('Real Estate Sector - Model Performance')

# accurate model accuracy
base_accuracy=r2_score(y_true=y_test, y_pred=preds_forest)
print('Accuracy Before = {:0.2f}%.'.format(100 *base_accuracy))

# improved accuracy
RegForestScore=grid.score(X_test, y_test)
print('Accuracy After = {:0.2f}%.'.format(100 *RegForestScore))

#percent of model improvement
print('Improvement of {:0.2f}%.'.format( 100 * (RegForestScore - base_accuracy) / base_accuracy))
```



2- Energy Sector (Grid Search)



```
param_grid = {
    "n_estimators": (50,100), # how many trees in our forest
    "max_depth": (0,100) # how deep each decision tree can be

}

grid = GridSearchCV(
    reg_forest_E,
    param_grid,
    cv = 5,
    n_jobs=-1,
    verbose=1
)

grid.fit(X_train_E, y_train_E)
```

```
# Model optimization evaluation
print('Energy Sector - Model Performance')

# accurate model accuracy
base_accuracy_E=r2_score(y_true=y_test_E, y_pred=preds_forest_E)
print('Accuracy Before = {:0.2f}%.'.format(100 *base_accuracy_E))

# improved accuracy
RegForestScore_E=grid.score(X_test_E, y_test_E)
print('Accuracy After = {:0.2f}%.'.format(100 *RegForestScore_E))

#percent of model improvement
print('Improvement of {:0.2f}%.'.format( 100 * (RegForestScore_E - base_accuracy_E) / base_accuracy_E))
```

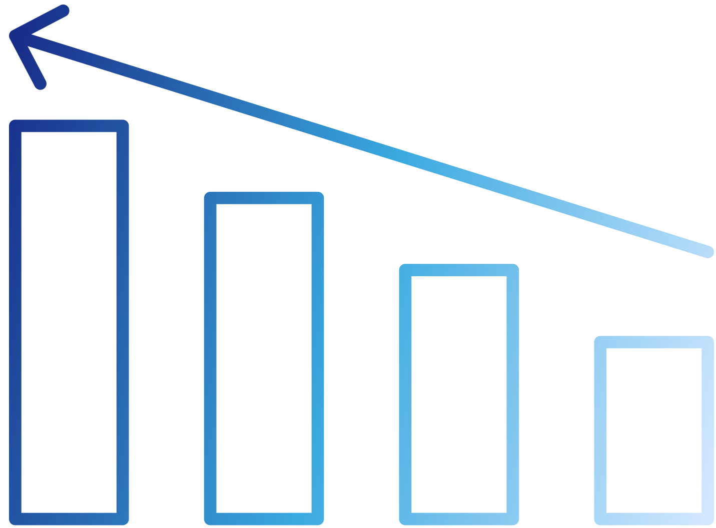
The Model Performance After the Grid Search

Real Estate Sector

```
Real Estate Sector - Model Performance
Accuracy Before = 93.60%.
Accuracy After = 94.46%.
Improvement of 0.93%.
```

Energy Sector

```
Energy Sector - Model Performance
Accuracy Before = 94.74%.
Accuracy After = 95.15%.
Improvement of 0.43%.
```



7. Model Pipeline

Real Estate Sector Pipeline

```python
# Create a transformer for numeric columns

numeric_transformer = Pipeline(
    steps=[

        ('imputer', SimpleImputer()),
        ('scaler', StandardScaler())
    ]
)
# Create a preprocessor transformer
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features)

    ]
)

# Append classifier to preprocessing pipeline.
# Now we have a full prediction pipeline.
clf = Pipeline(
    steps=[
        ('preprocessor', preprocessor),
        ('classifier',RandomForestRegressor(n_estimators = 10, random_state = 0, criterion = 'mse'))
    ]
)
```

Real Estate Sector Pipeline

Real Estate Sector Accuracy After pipeline

```
clf.fit(X_train, y_train)
 # Find out the pipline score
RE_score = clf.score(X_test, y_test)*100

# display the score with 2 digits only
RegTreeScore=float('{0:.2f}'.format(RE_score))

print(f"Model Score :",RegTreeScore)
```

```
Model Score : 93.52
```

Energy Sector Pipeline

```
# Create a transformer for numeric columns

numeric_transformer = Pipeline(
    steps=[

        ('imputer', SimpleImputer()),
        ('scaler', StandardScaler())
    ]
)


# Create a preprocessor transformer
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features)

    ]
)

# Append classifier to preprocessing pipeline.
# Now we have a full prediction pipeline.
clf = Pipeline(
    steps=[
        ('preprocessor', preprocessor),
        ('classifier',RandomForestRegressor(n_estimators = 10, random_state = 0, criterion = 'mse'))
    ]
)
```

Energy Sector Pipeline

Energy Sector Accuracy After pipeline

```
clf.fit(X_train_E, y_train_E)

 # Find out the pipline score
RE_score_E = clf.score(X_test_E, y_test_E)*100

# display the score with 2 digits only
RegTreeScore_E=float('{0:.2f}'.format(RE_score_E))

print(f"Model Score :",RegTreeScore_E)
```

```
Model Score : 94.75
```

Conclusion

Thank you for listening!