CAPSTONE PROJECT

ANALYZING SAUDI STOCK EXCHANGE (TADAWUL) WITH MACHINE LEARNING

DAQUEST GROUP

CONTENT

INTRODUCTION

- Saudi’s 2030 Vision and Tadawul
- Business problem
- Project goal
  - Predicts the close price
  - Predict the stock change

INTRODUCTION

Data Review

TADAWAL DATASET

- This is the data of Saudi stock market companies from 2001–12–31 until 2020–04–16.
- Collected from Saudi Stock Exchange (Tadawul) website.
- This dataset has 14 columns and 593819 rows.

DATA REVIEW

TADAWAL DATASET

DATA REVIEW

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
(sectoer (String	The sector in which the company operates.
(date (Date	The date of the stock price.
(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.

Preprocessing Dataset

Presented By : Jawhara Almulhim

TADAWAL DATASET

DATA PREPROCESSING

- Data set shape

```
# Check the shape of  dataset
tadawul_stuks.shape
```

```
(593819, 14)
```

- Change the column Name

```
change_names = {'name': 'Company_name','sectoer':'sector' , 'open':'open_price' , 'high': 'high_price' ,'low' :'low_price',
               'close' : 'close_price', 'trading_name ': 'trading_name',
                'volume_traded ': 'volume_traded','no_trades ':'num_trades'}

# Place them on our dataframe
tadawul_stuks.rename(columns=change_names, inplace= True)
```

DATA PREPROCESSING

DATA PREPROCESSING

- Add year, month and day columns

```
tadawul_stuks["Year"] = pd.DatetimeIndex(tadawul_stuks['date']).year
tadawul_stuks["month"] = pd.DatetimeIndex(tadawul_stuks['date']).month_name()
tadawul_stuks["day"] = pd.DatetimeIndex(tadawul_stuks['date']).day_name()
```

- Drop unnecessary columns

```
#remove unnecessary column symbol, company_name
tadawul_stuks.drop('symbol', inplace=True, axis=1)
tadawul_stuks.drop('Company_name', inplace=True, axis=1)
```

- Add the categorical target column

```
tadawul_stuks.loc[tadawul_stuks['perc_Change'] > 0.00 , "Change_category"] = 'Good Change'
tadawul_stuks.loc[tadawul_stuks['perc_Change'] <= -0.00 , "Change_category"] = 'Bad Change'
tadawul_stuks.loc[tadawul_stuks['perc_Change'] == 0.00 , "Change_category"] = 'Stable'
```

DATA CLEANING

DATA PREPROCESSING

- Check the null values

```
tadawul_stuks.isnull().sum()
```

```
symbol                 0
Company_name           0
trading_name           0
sector                 0
date                   0
open_price          6455
high_price          6697
low_price           6697
close_price            0
change                 0
perc_Change            0
volume_traded          0
value_traded           0
num_trades          7691
Year                   0
month                  0
day                    0
Change_category        0
dtype: int64
```

- Handle the null values

```
tadawul_stuks = tadawul_stuks.dropna()
```

DATA CLEANING

- **Check for null values again**

```
# Check for null values again
tadawul_stuks.isnull().sum()
```

```
trading_name        0
sector              0
date                0
open_price          0
high_price          0
low_price           0
close_price         0
change              0
perc_Change         0
volume_traded       0
value_traded        0
num_trades          0
Year                0
month               0
day                 0
Change_category     0
dtype: int64
```

- **Check the duplicate data**

```
tadawul_stuks.duplicated().any()
```

```
False
```

DATA CLEANING

DATA PREPROCESSING

- Change the name of the sector to make it short before plotting

```
tadawul_stuks['sector'] = tadawul_stuks['sector'].replace('Information Technology', 'IT')
```

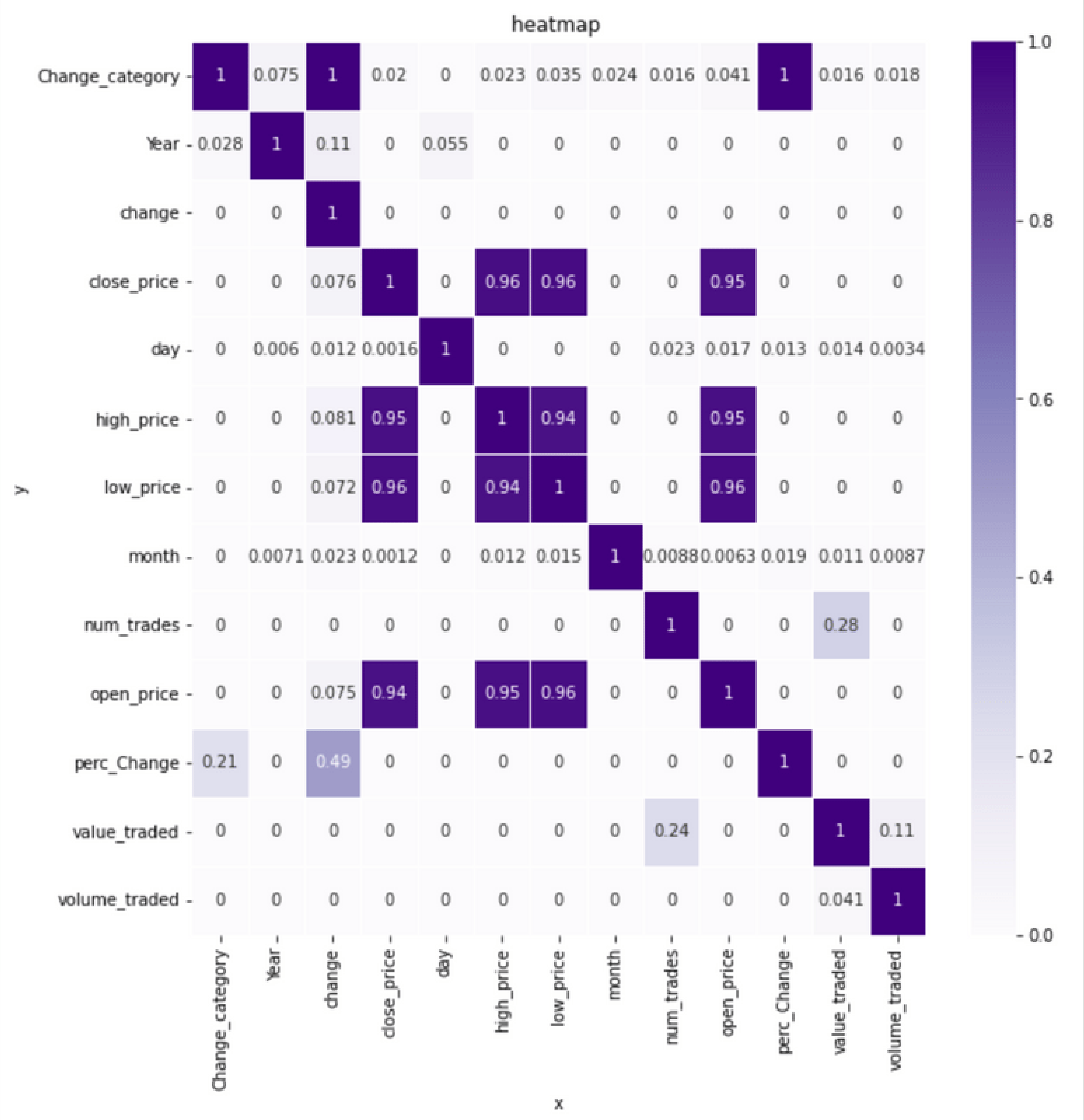
- Check the shape and size after preprocessing

```
# Check the shape of  dataset
tadawul_stuks.shape
```

```
(579431, 16)
```

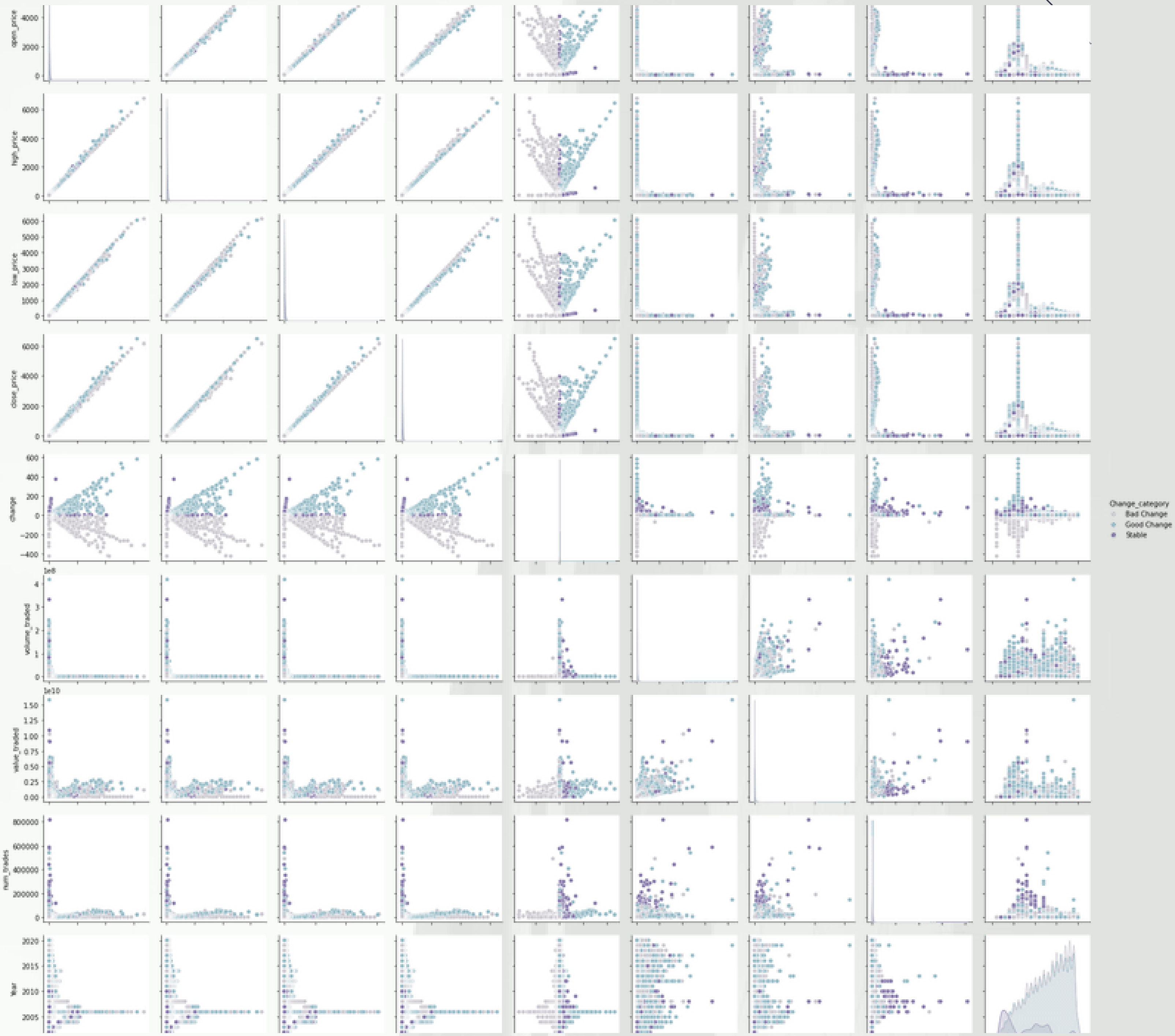
Exploratory Data Analysis (EDA)

PLOT 1

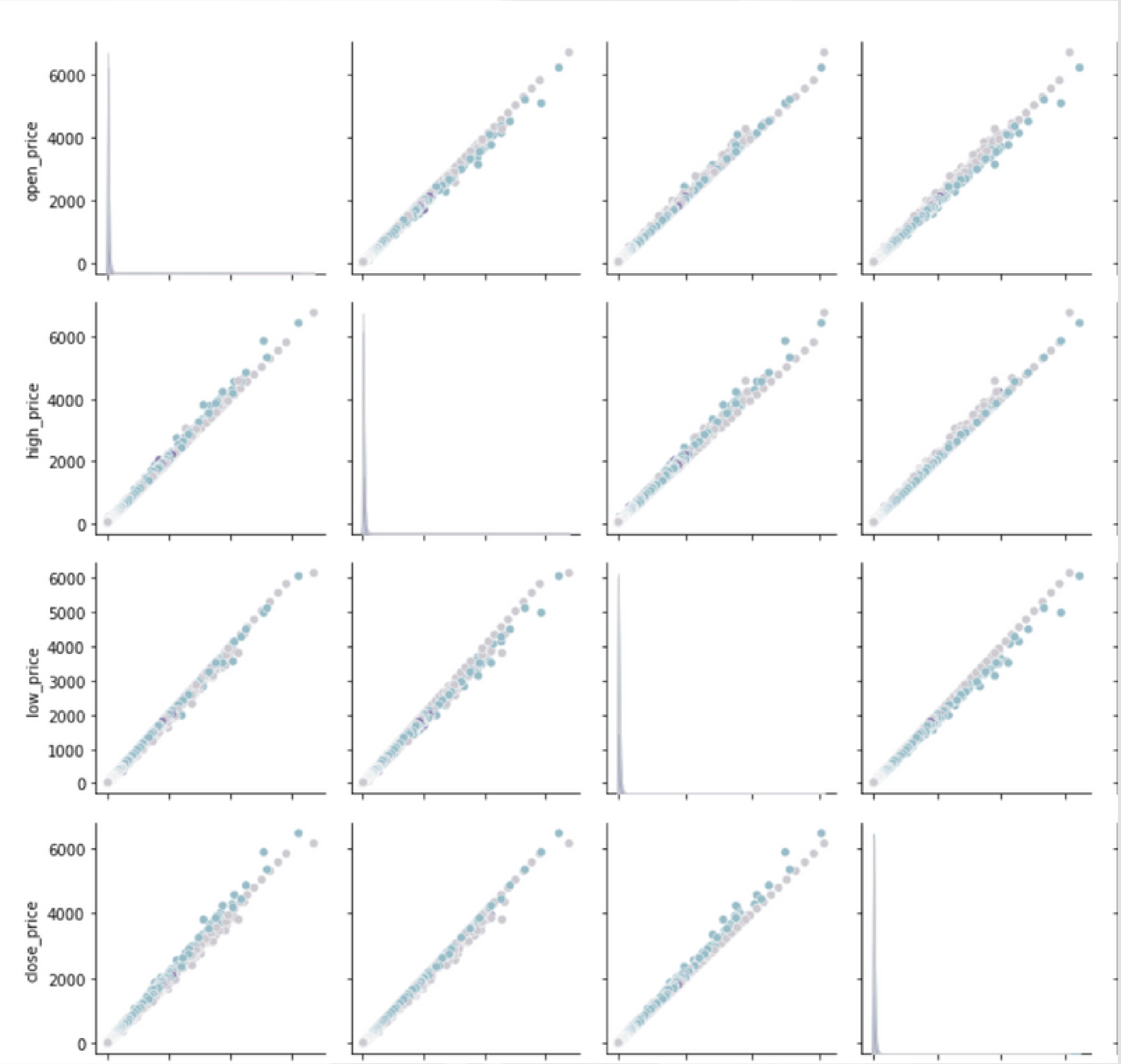


heatmap
Change_category
Year
change
close_price
day
high_price
low_price
month
num_trades
open_price
perc_Change
value_traded
volume_traded
x
y

PLOT 2

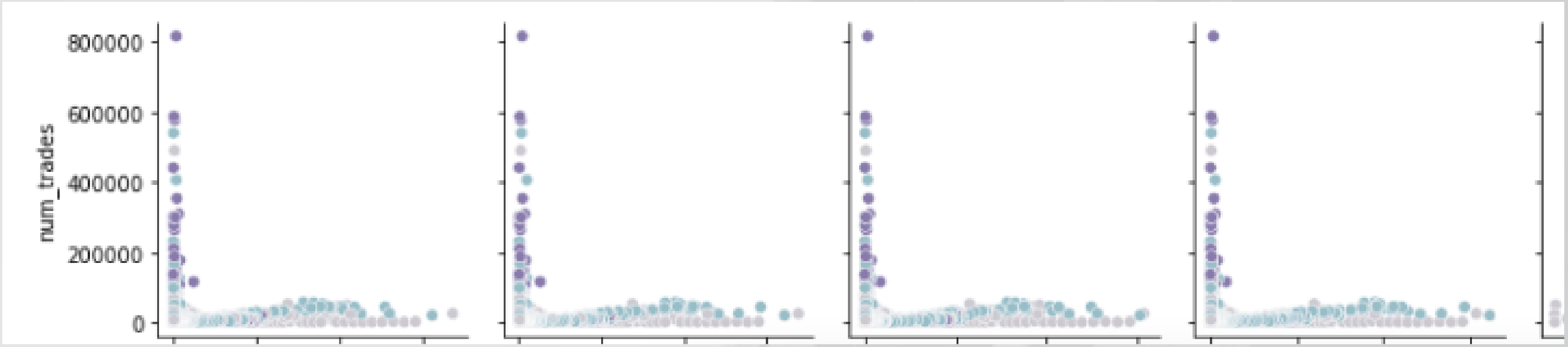

open_price
high_price
low_price
close_price
change
volume_traded
value_traded
num_trades
Year
Change_category
Bad Change
Good Change
Stable

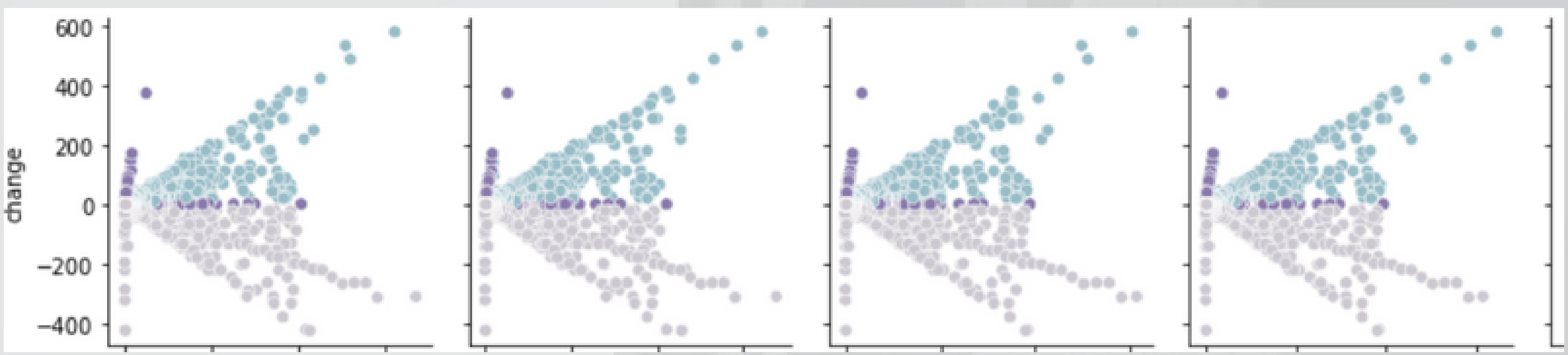
PLOT 2

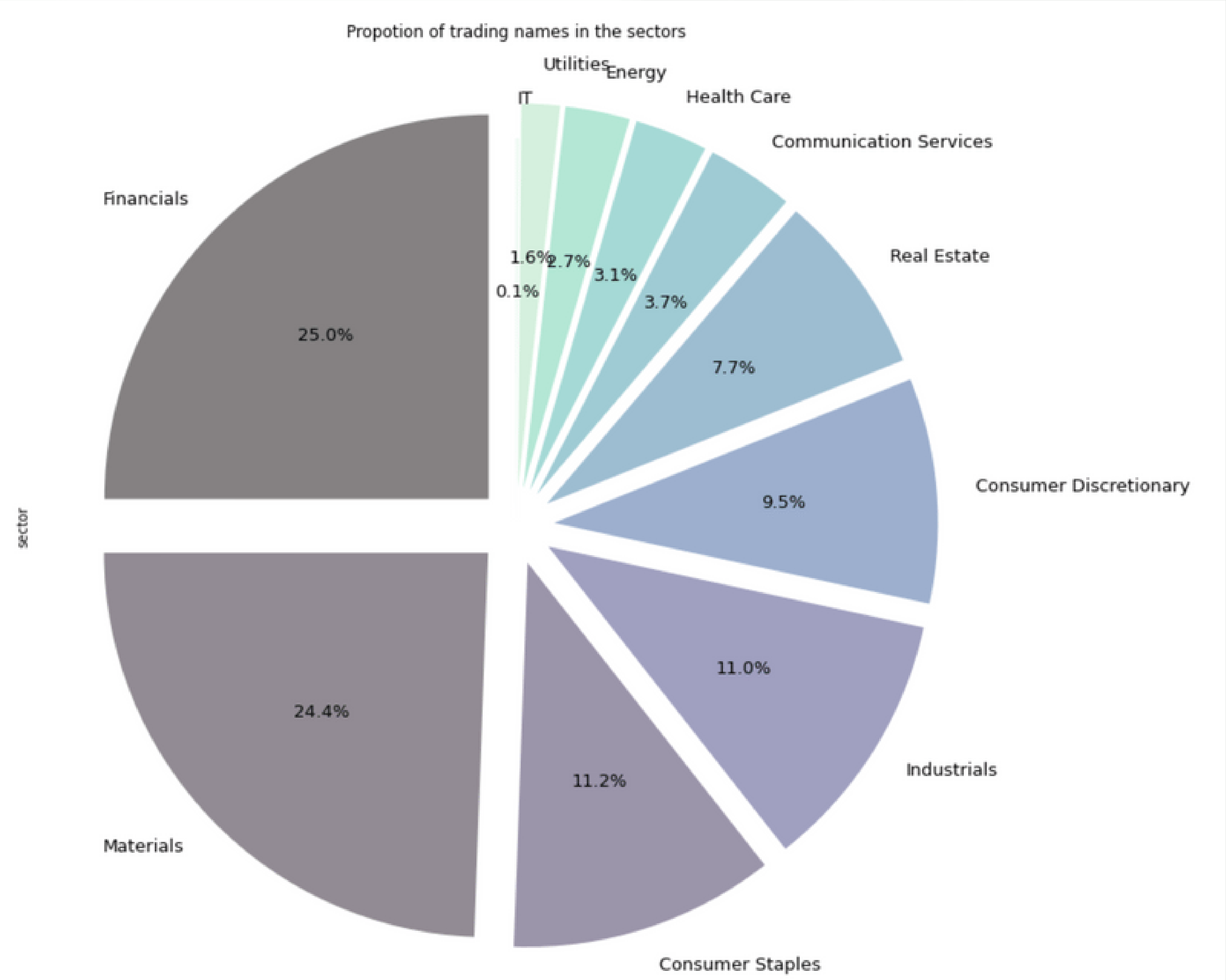


open_price
high_price
low_price
close_price
6000
4000
2000
0
5000
3000
1000

PLOT 2


num_trades
800000
600000
400000
200000
0

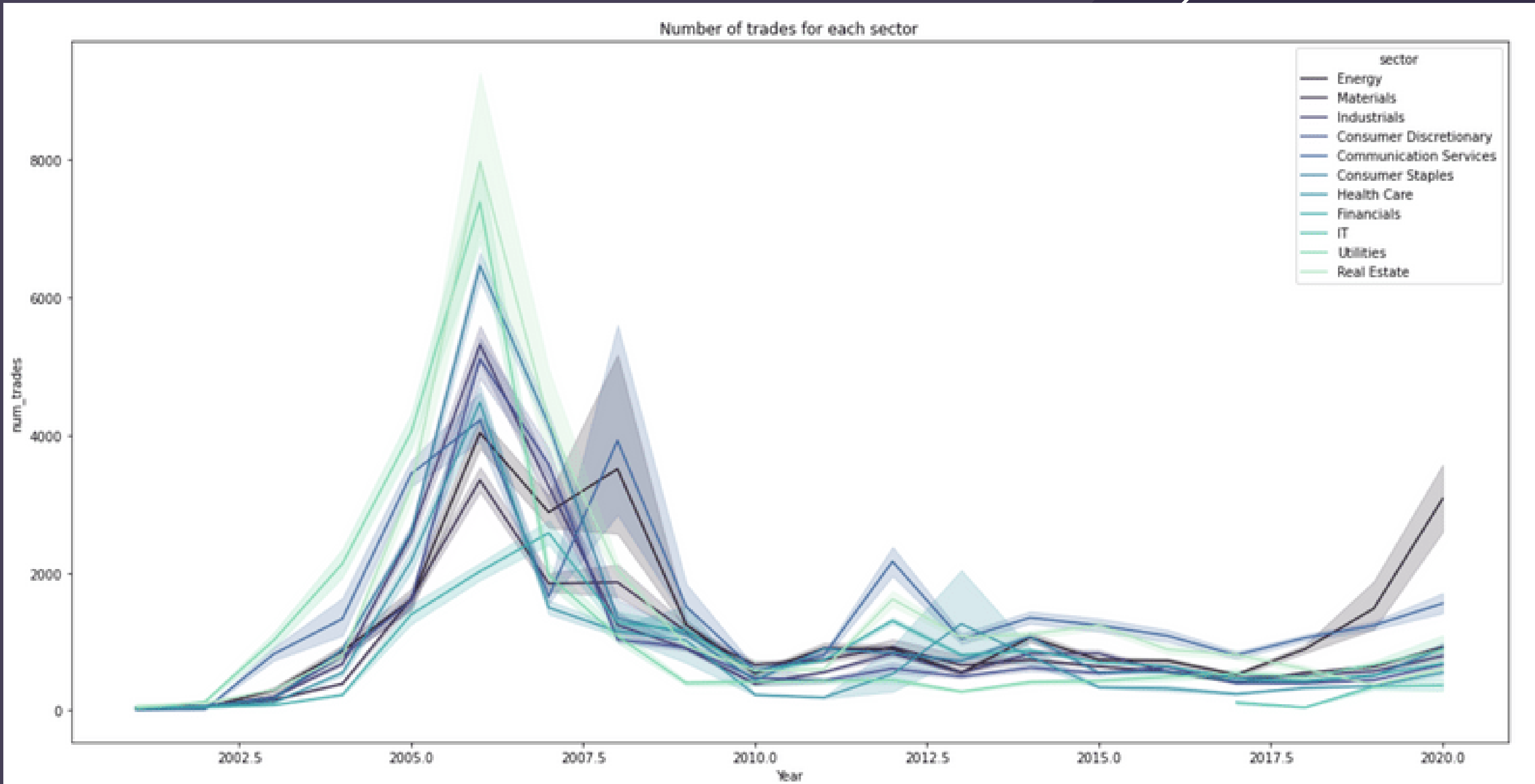

change
600
400
200
0
−200
−400



Propotion of trading names in the sectors
Utilities
Energy
IT
Health Care
Communication Services
Financials
Real Estate
Consumer Discretionary
Industrials
Materials
Consumer Staples
sector
1.6%
2.7%
0.1%
3.1%
3.7%
25.0%
7.7%
9.5%
11.0%
24.4%
11.2%

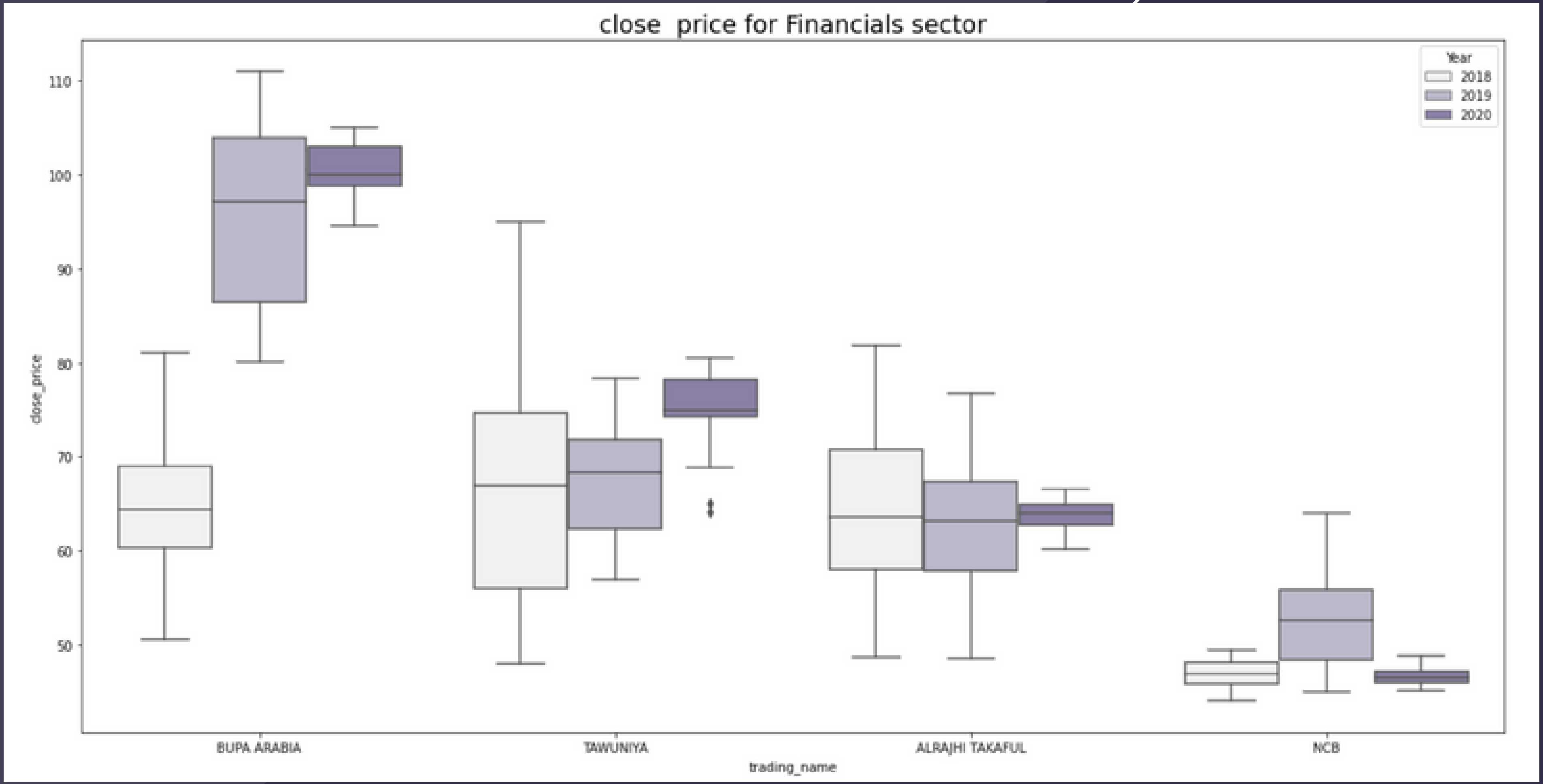
PLOT 3

PLOT 4

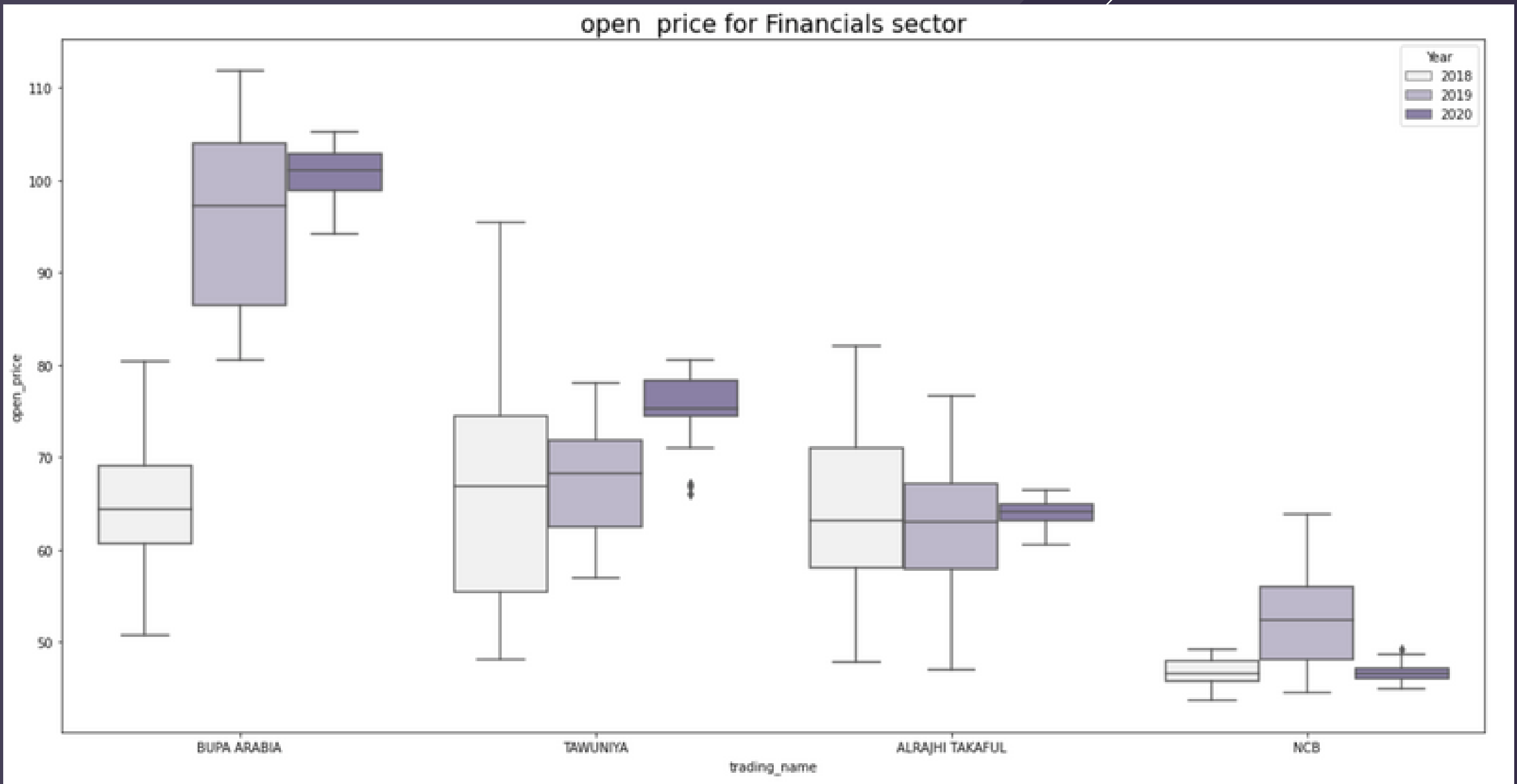


Number of trades for each sector
num_trades
Year
0
2000
4000
6000
8000
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5
2020.0
sector
Energy
Materials
Industrials
Consumer Discretionary
Communication Services
Consumer Staples
Health Care
Financials
IT
Utilities
Real Estate

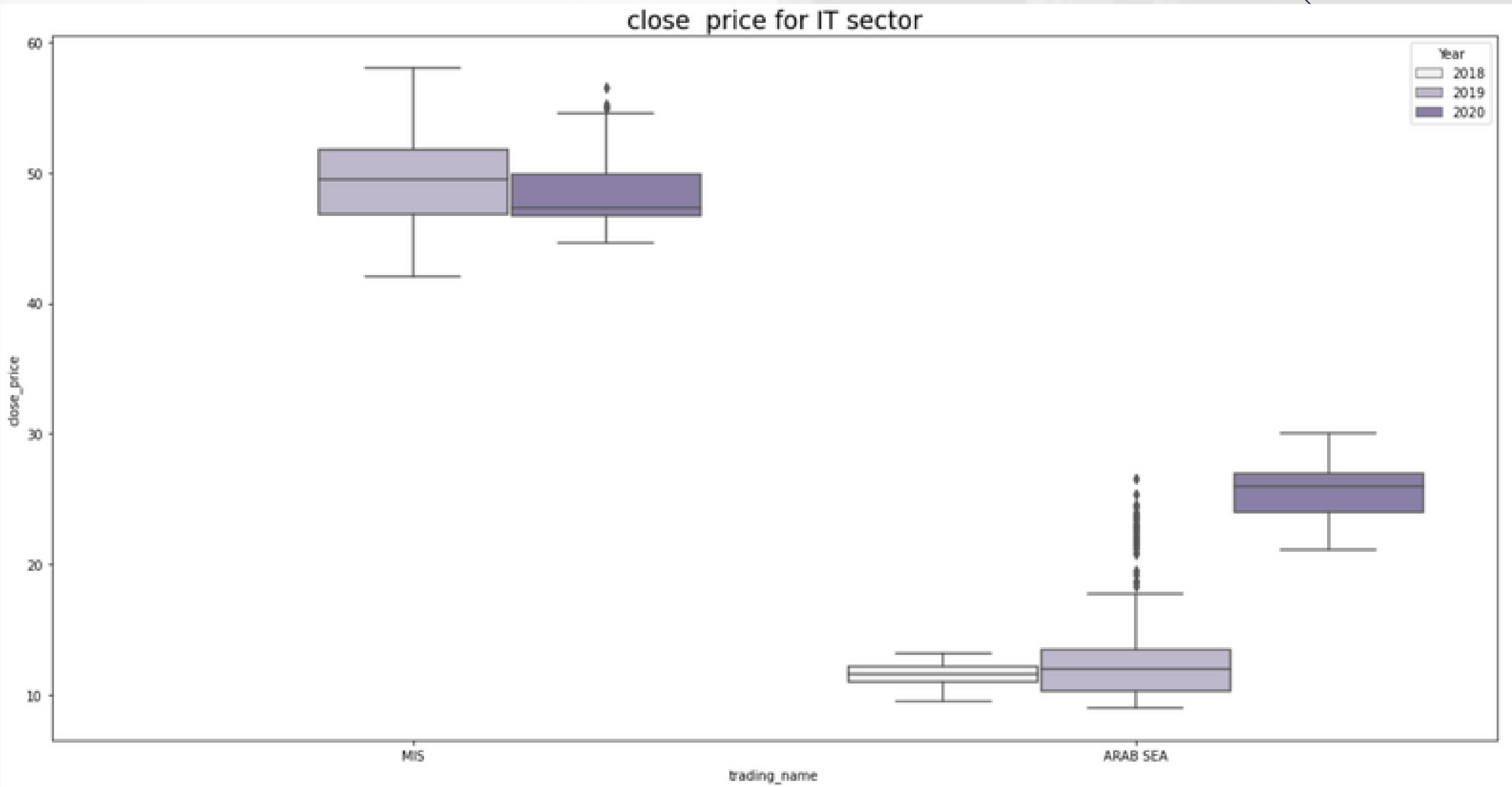
PLOT 5


close price for Financials sector
Year
2018
2019
2020
110
100
90
80
70
60
50
close_price
BUPA ARABIA
TAWUNIYA
ALRAJHI TAKAFUL
NCB
trading_name

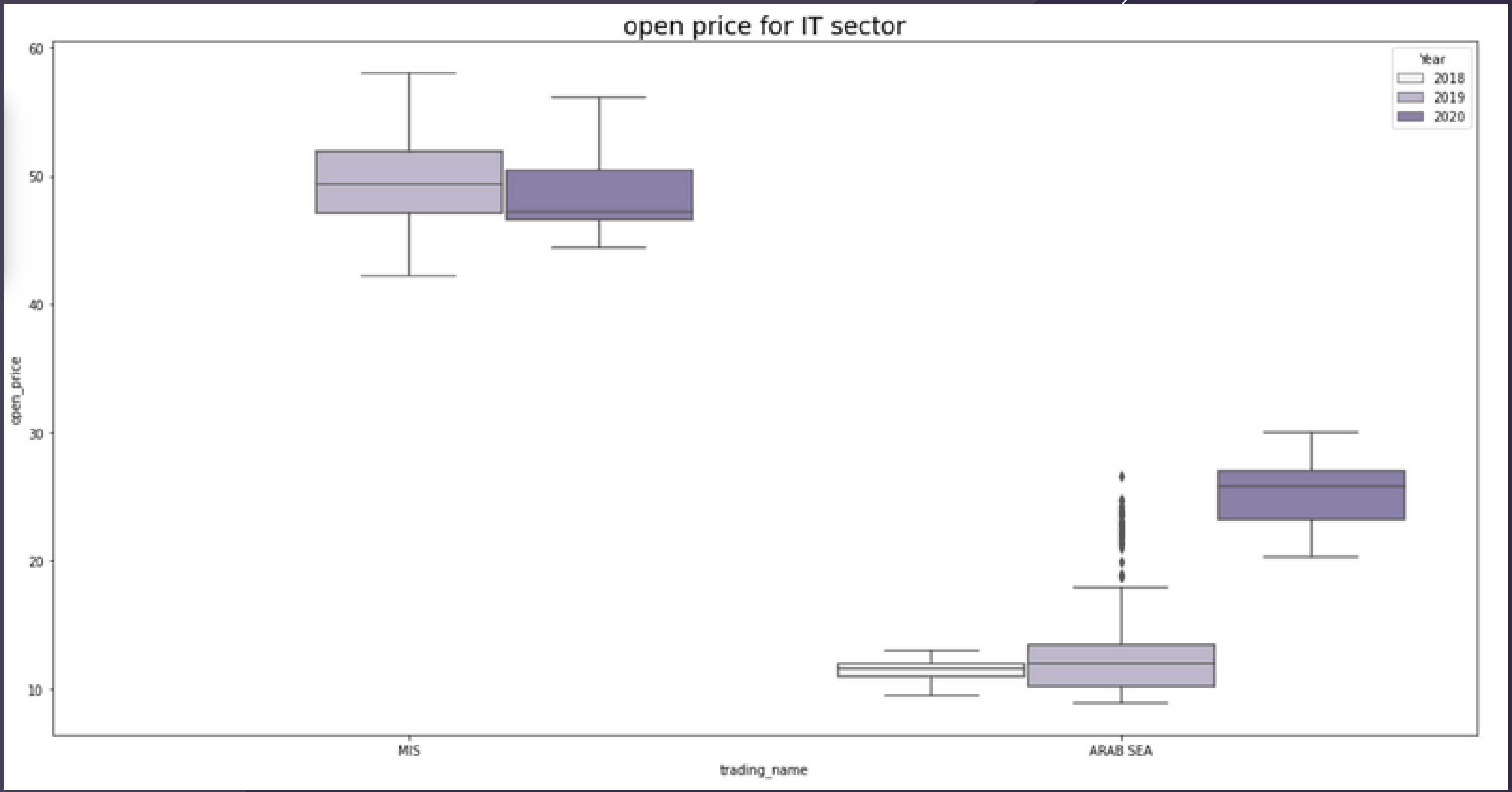
PLOT 5


open price for Financials sector
Year
2018
2019
2020
110
100
90
80
70
60
50
open_price
BUPA ARABIA
TAWUNIYA
ALRAJHI TAKAFUL
NCB
trading_name

PLOT 6

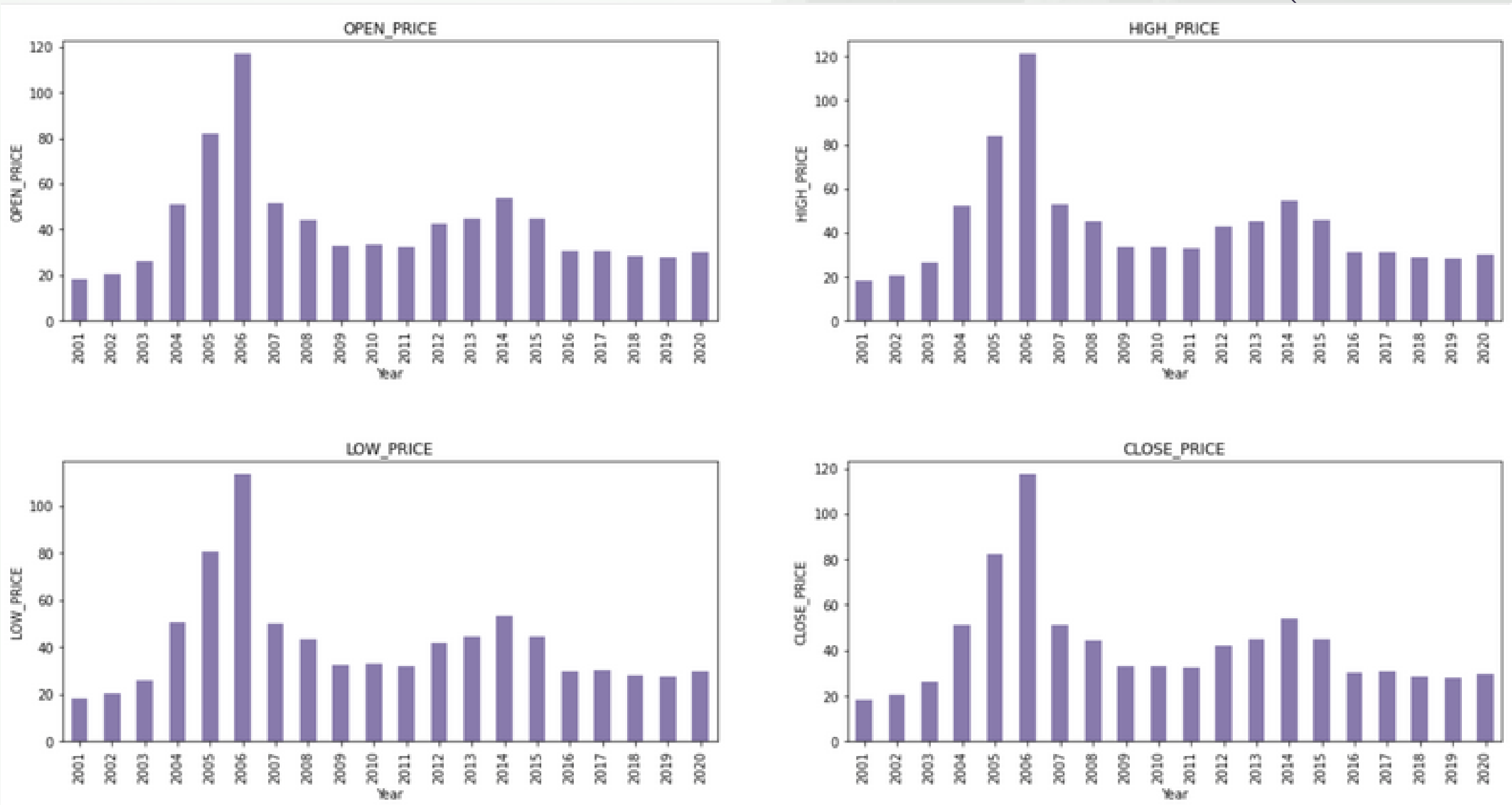


close price for IT sector
Year
2018
2019
2020
60
50
40
30
20
10
close_price
MIS
ARAB SEA
trading_name

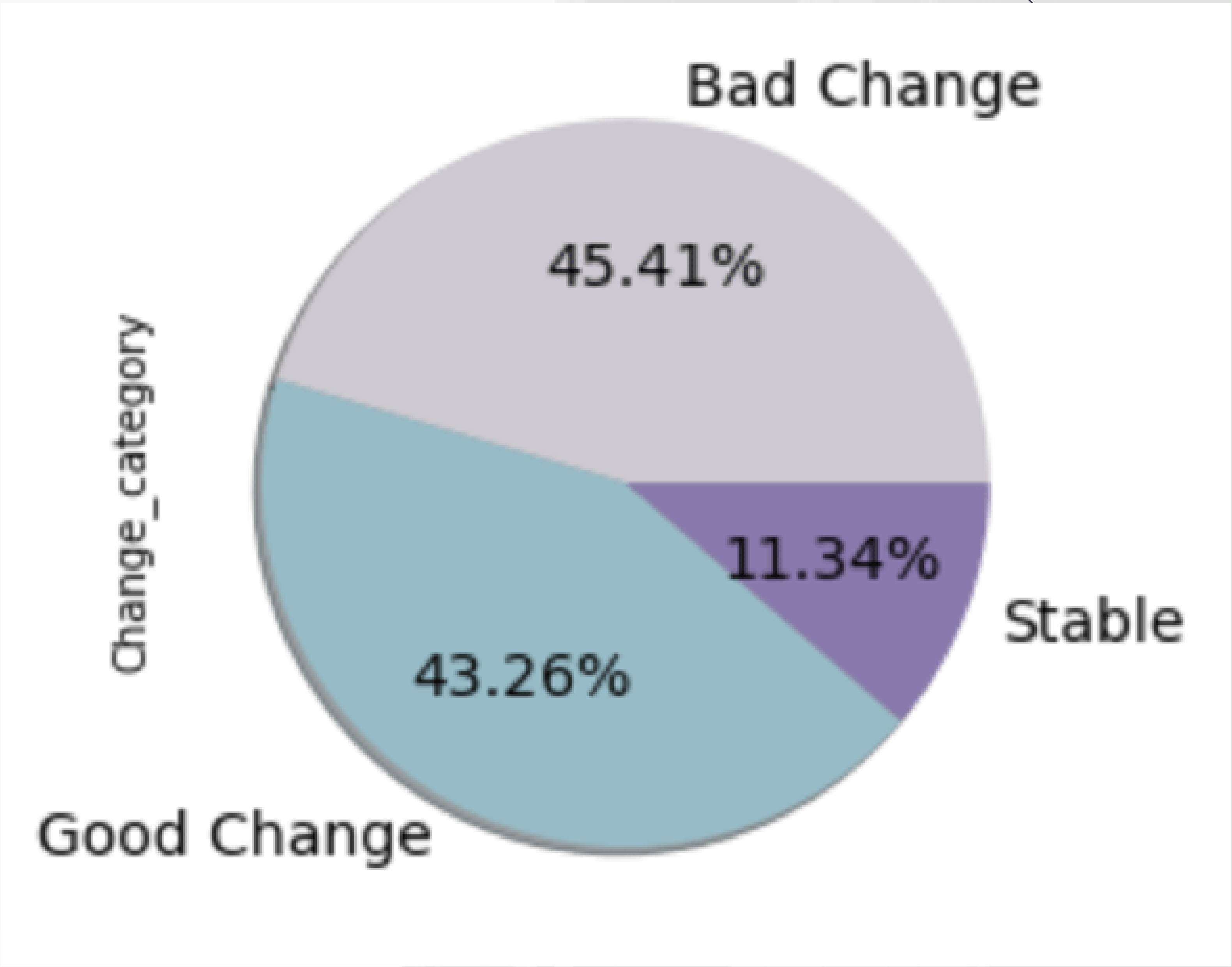

open price for IT sector
Year
2018
2019
2020
open_price
60
50
40
30
20
10
MIS
ARAB SEA
trading_name

PLOT 6

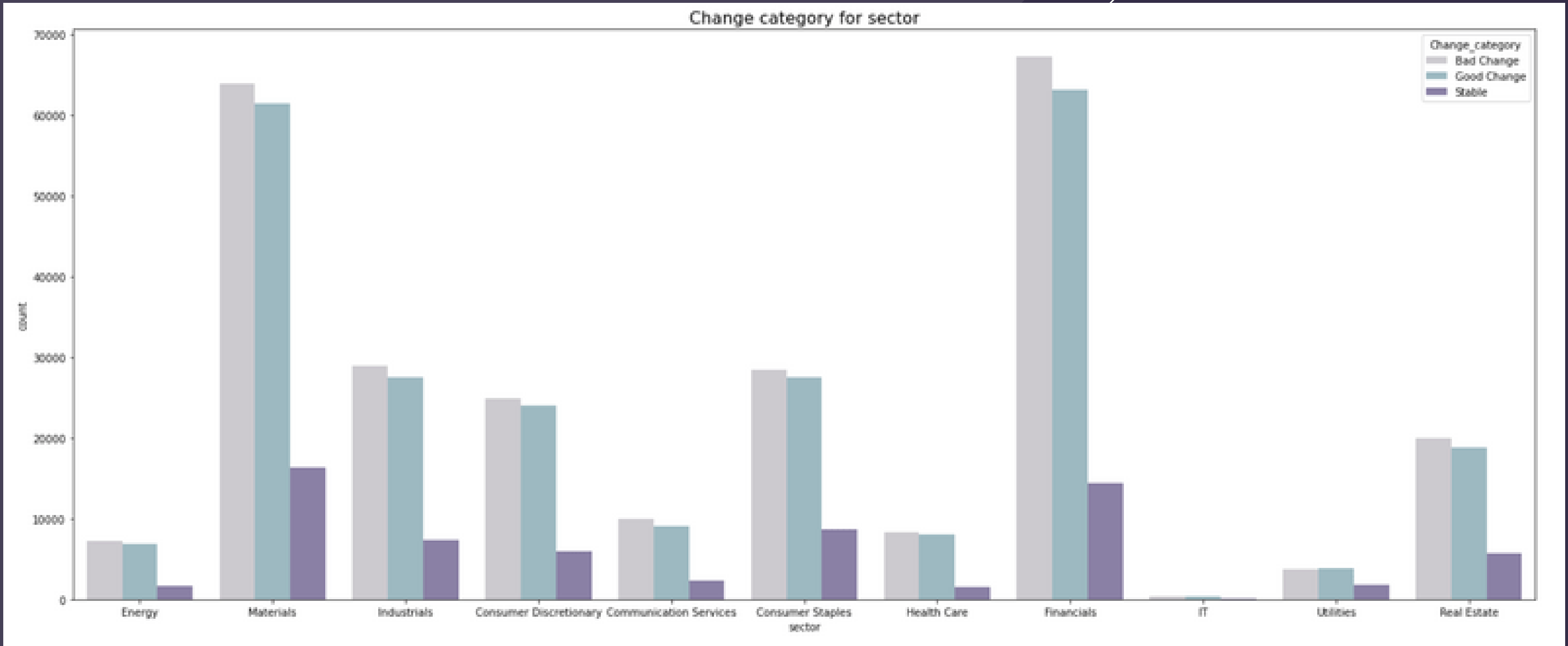
PLOT 7


OPEN_PRICE
HIGH_PRICE
LOW_PRICE
CLOSE_PRICE
Year
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
0
20
40
60
80
100
120

PLOT 8

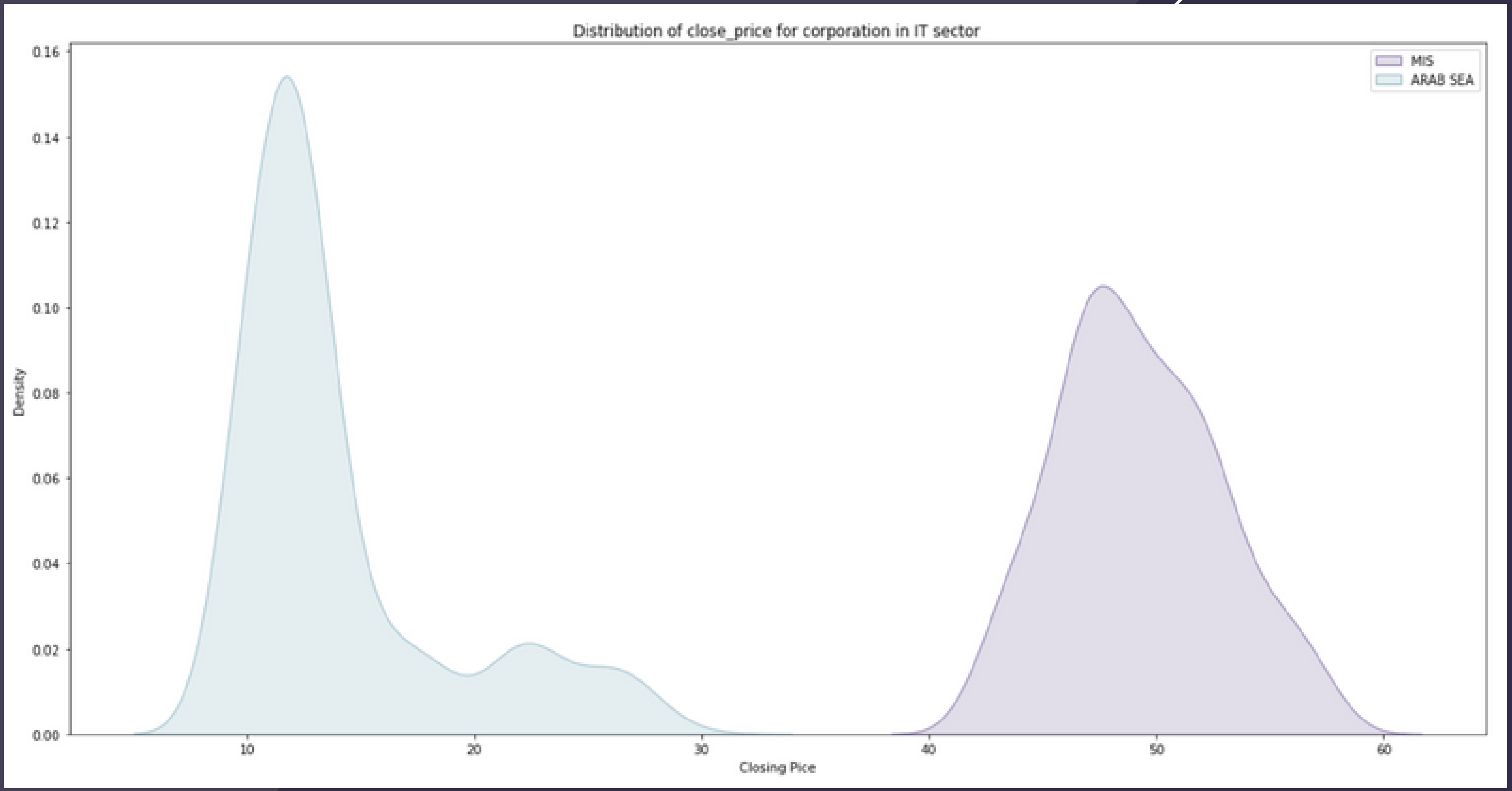

Bad Change
45.41%
Change_category
11.34%
Stable
43.26%
Good Change

PLOT 9



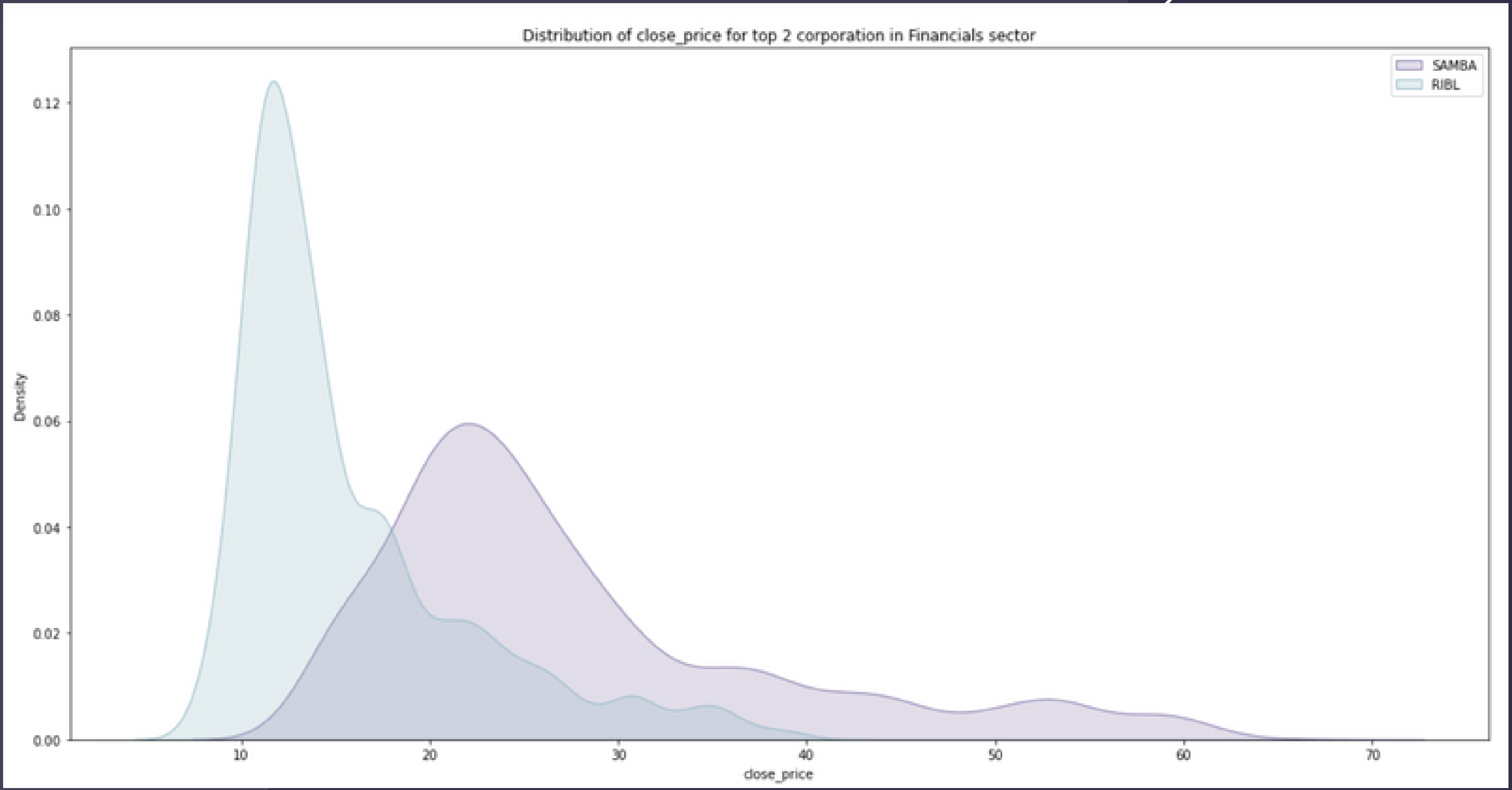
Change category for sector
count
sector
Change_category
Bad Change
Good Change
Stable
Energy
Materials
Industrials
Consumer Discretionary
Communication Services
Consumer Staples
Health Care
Financials
IT
Utilities
Real Estate
0
10000
20000
30000
40000
50000
60000
70000

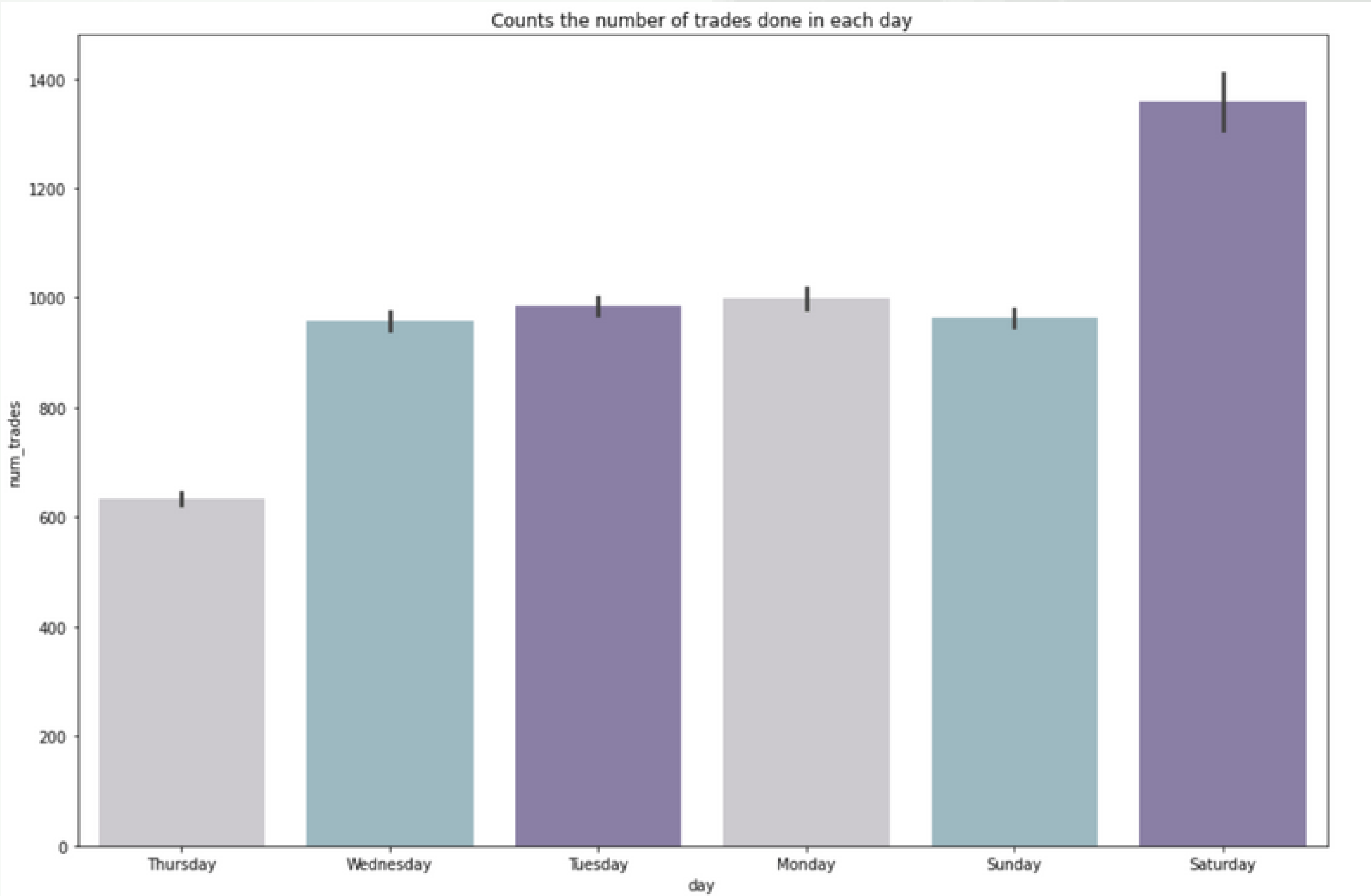
PLOT 10



Distribution of close_price for corporation in IT sector
MIS
ARAB SEA
Density
Closing Pice
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
10
20
30
40
50
60

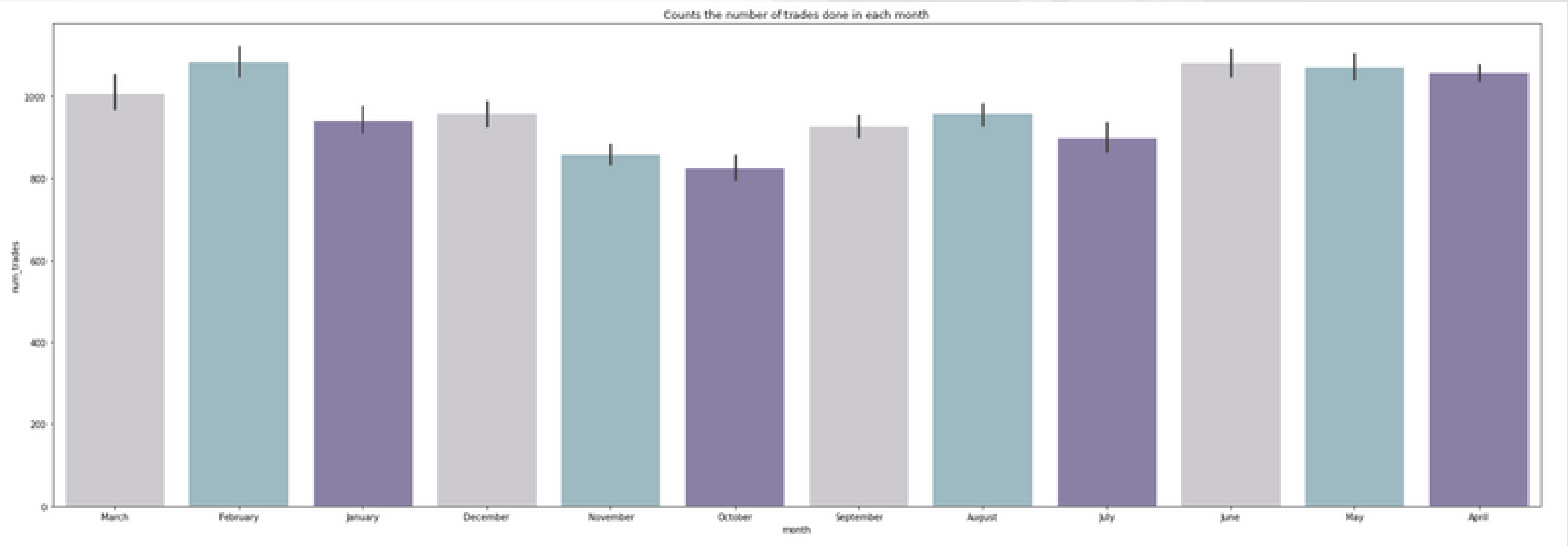
PLOT 10


Distribution of close_price for top 2 corporation in Financials sector
SAMBA
RIBL
Density
close_price
0.00
0.02
0.04
0.06
0.08
0.10
0.12
10
20
30
40
50
60
70


Counts the number of trades done in each day
1400
1200
1000
800
600
400
200
0
num_trades
Thursday
Wednesday
Tuesday
Monday
Sunday
Saturday
day

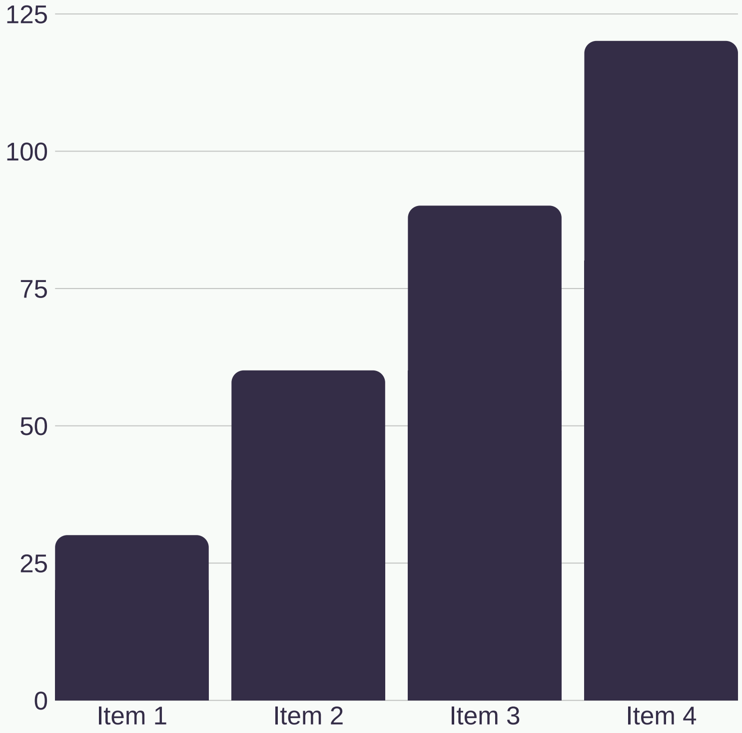
PLOT 11

PLOT 11

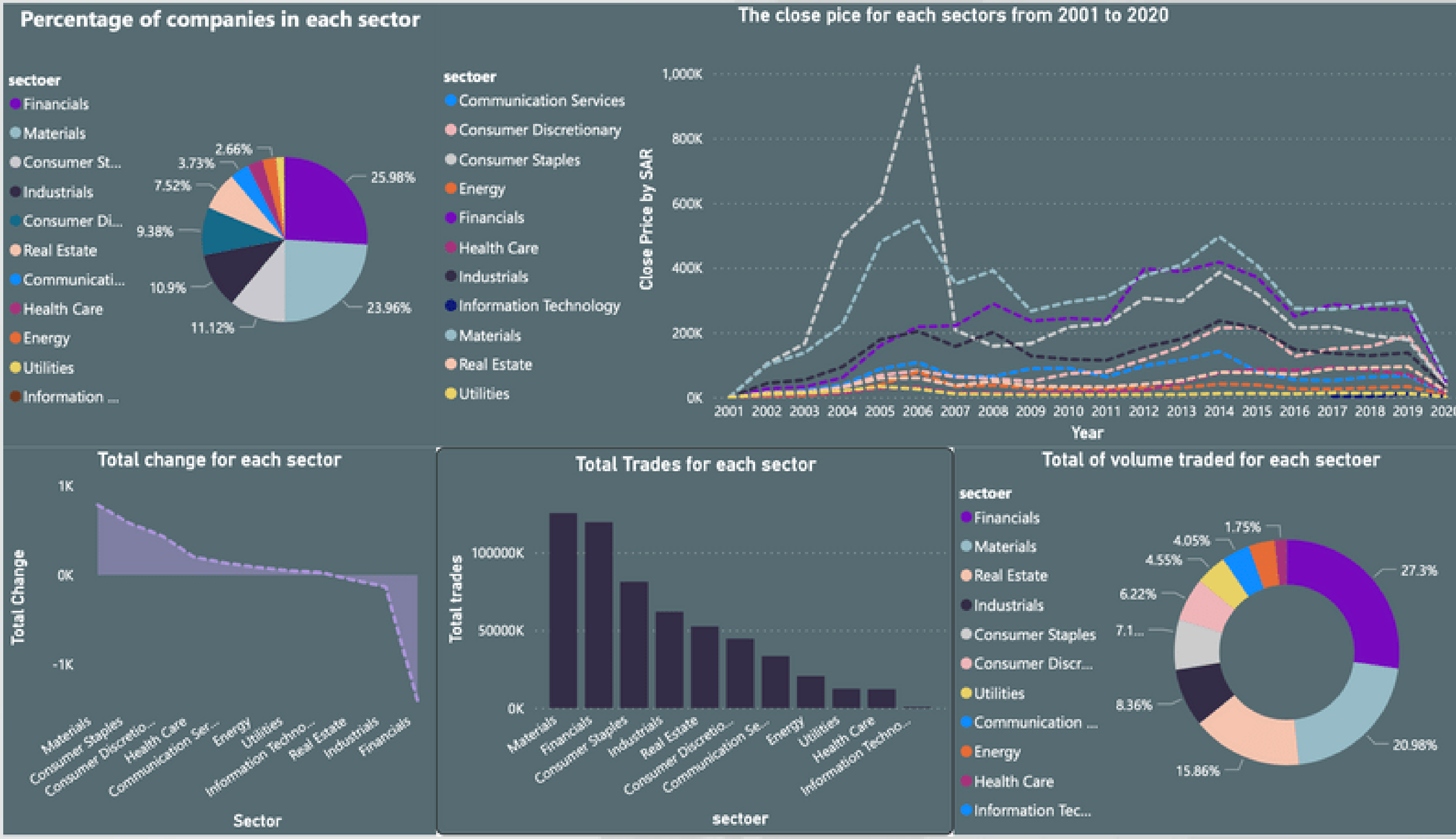


Presented By :Maha Alharbi

Dashboards



DASHBOARD


Percentage of companies in each sector
sectoer
Financials
Materials
Consumer St...
Industrials
Consumer Di...
Real Estate
Communicati...
Health Care
Energy
Utilities
Information ...
2.66%
3.73%
7.52%
9.38%
10.9%
11.12%
25.98%
23.96%
The close pice for each sectors from 2001 to 2020
sectoer
Communication Services
Consumer Discretionary
Consumer Staples
Energy
Financials
Health Care
Industrials
Information Technology
Materials
Real Estate
Utilities
Close Price by SAR
1,000K
800K
600K
400K
200K
0K
2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020
Year
Total change for each sector
Total Change
1K
0K
-1K
Materials
Consumer Staples
Consumer Discretio...
Health Care
Communication Ser...
Energy
Utilities
Information Techno...
Real Estate
Industrials
Financials
Sector
Total Trades for each sector
Total trades
100000K
50000K
0K
Materials
Financials
Consumer Staples
Industrials
Real Estate
Consumer Discretio...
Communication Se...
Energy
Utilities
Health Care
Information Techno...
sectoer
Total of volume traded for each sectoer
sectoer
Financials
Materials
Real Estate
Industrials
Consumer Staples
Consumer Discr...
Utilities
Communication ...
Energy
Health Care
Information Tec...
1.75%
4.05%
4.55%
6.22%
7.1...
8.36%
15.86%
27.3%
20.98%

MachineLearning Models

- Regression Model
- Classifications Model

Presented By : Samar Alosaimi

REGRESSION MODELS

SELECT THE FEATURE AND TARGET

```
target = 'close_price' # Target Varible
features = ['open_price','low_price','change']
```

SCALER

```
scaler = StandardScaler()
scaled_df = scaler.fit_transform(X)
```

SPLIT THE DATA

```
# split data into train and test
# select random state = 42
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
```

BASELINE MODEL

```
# we will use the DummyRegressor model for the baseline
baseline_model = DummyRegressor()
baseline_model.fit(X_train,y_train)
baseline_model_pred =  baseline_model.predict(X_test)

print(f"baseline model score: {r2_score(y_test, baseline_model_pred)}")
```

BASELINE MODEL RESULT

	Model	R2	MSE	MAE	RMSE
0	Baseline model	-0.000013	7217.246782	28.401061	84.954381

LINER REGRESSION

```
#create the model
liner = LinearRegression()
# fit the model using X train and y train
liner.fit(X_train , y_train)
# using X test to make our predication
linear_pred = liner.predict(X_test)
```

MODELS EVALUATION

```
# Create cost function that display all the cost functions for the regression models
def cost_function(pred):
    Adj_r2 = 1 - (1-r2_score(y_test, pred)) * (len(y_test)-1)/(len(y_test)-X_test.shape[1]-1)
    print("R Squared:",r2_score(y_test, pred))
    print("MSE:",mean_squared_error(y_test, pred))
    print("MAE:",mean_absolute_error(y_test, pred))
    print("RMSE:",np.sqrt(mean_squared_error(y_test, pred)))
    print("Adjusted R Squared:",Adj_r2)
```

LINER REGRESSION RESULT

	Model	R2	MSE	MAE	RMSE
1	Linear Regression	0.999606	2.844279	0.426416	1.686499

REGRESSION MODELS

MODELS RESULTS

	Model	R2	MSE	MAE	RMSE
0	Baseline model	-0.000013	7217.246782	28.401061	84.954381
1	Linear Regression	0.999606	2.844279	0.426416	1.686499
2	Random Forest	0.987333	91.422137	5.789177	9.561492
3	KNN	0.999515	3.501239	0.301051	1.871160
4	GBR	0.999223	5.608102	0.671724	2.368143
5	XGB	0.999213	5.676292	0.396874	2.382497

MODEL OPTIMIZATION - HYPERPARAMETER TUNING

```
param_linear = {"fit_intercept": [True, False]}

random_linear_reg = RandomizedSearchCV(liner,# set the model
                                       param_linear,# set the parameter
                                       scoring='r2',
                                       verbose=1,
                                       n_jobs=-1)

# fit the model
random_linear_reg.fit(X_train, y_train)
# make the predication
random = random_linear_reg.predict(X_test)
```

LINER REGRESSION AFTER TUNING RESULT

	Model	R2	MSE	MAE	RMSE
2	Linear Regression with tuning	0.999606	2.844279	0.426416	1.686499

REGRESSION MODELS

PIPELINE FOR BEST MODEL - LINER REGRESSION

```
numeric_features = X_train.describe().columns # Select the numrical feature

#Create Transformer for numerical data
numeric_transformer = Pipeline(
    steps=[
        ('imputer', SimpleImputer(strategy="most_frequent")),
        ('scaler', StandardScaler())
    ]
)
# Create a preprocessor transformer
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
    ]
)
# create the pipline
liner_reg_pp = Pipeline(
    steps=[
        ('preprocessor', preprocessor),# set the preprocessor

        ('reg',LinearRegression())# Create the Liner regression model
    ]
)
```

MODELS SUMMARY

-The best performance is the linear regression model followed by knn model .

-The worst performance is the Random forest regressor.

- Select the linear regression as the best model depend on the lowest RMSE value.

Machine Learning Models

- Regression Model
- Classifications Model

Presented By : Fatima Almulhim

REGRESSION MODELS

SELECT THE FEATURE AND TARGET

```
# set our features
cal_fatures= ['open_price', 'close_price', 'high_price', 'low_price']

#Set our target which is the the close price
cal_target= ['Change_category']

X= df[cal_fatures]
y= df[cal_target]
```

SCALER

```
scaler = StandardScaler()
scaled_df = scaler.fit_transform(X)
```

SPLIT THE DATA

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 42)
```

BASELINE MODEL

```
dummy1 = DummyClassifier()
dummy1.fit(X_train, y_train)
y_pred_dum = dummy1.predict(X_test)
```

BASELINE MODEL RESULT

	Model	Accuracy	Recall	precision	F1 score
0	Baseline model	0.454031	0.206144	0.454031	0.283548

CLASSIFICATION MODELS

SELECT THE FEATURE AND TARGET

```
# set our features
cal_fatures= ['open_price', 'close_price', 'high_price', 'low_price']

#Set our target which is the the close price
cal_target= ['Change_category']
```

RANDOM FOREST

```
# Create and fit the model
clas_forest = RandomForestClassifier(n_estimators = 100, criterion = 'gini', random_state = 42)
clas_forest.fit(X_train, y_train)
# Make orediction besed on the test set
preds_ran = clas_forest.predict(X_test)
```

RANDOM FOREST RESULT

```
# Print the classification report
print(classification_report(y_test, preds_ran))
```

```
              precision    recall  f1-score   support

  Bad Change       0.78      0.81      0.80     65770
 Good Change       0.78      0.81      0.79     62655
      Stable       0.49      0.36      0.41     16433

    accuracy                           0.76    144858
   macro avg       0.69      0.66      0.67    144858
weighted avg       0.75      0.76      0.75    144858
```

MODELS RESULTS

	Model	Accuracy	Recall	precision	F1 score
0	Baseline model	0.454031	0.206144	0.454031	0.283548
1	Logistic Regression	0.758729	0.741616	0.758729	0.713941
2	Random Forest	0.758529	0.749616	0.758529	0.752706
3	KNN	0.756320	0.744912	0.756320	0.744185
4	GBC	0.693741	0.694983	0.693741	0.666824

MODEL OPTIMIZATION - HYPERPARAMETER TUNING

```
parameters ={'bootstrap': [True, False],
 'max_depth': [10, 20, 30, None],
 'max_features': ['auto', 'sqrt'],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [50, 80, 100]}


random_search = RandomizedSearchCV(clas_forest, # Model
                    parameters, # Parameters to tune
                    cv=5, # Cross Validation
                    verbose=1, # Shows output while training
                    n_jobs=-1, # How many core to use on your computer (-1 means use all cores)
                    scoring="accuracy"
                   )
```

```
# Fit the model
random_search.fit(X_train, y_train)
```

LINER REGRESSION AFTER TUNING RESULT

	Model	Accuracy	Recall	precision	F1 score
2	Random Forest with tuning	0.750956	0.758492	0.768822	0.760486

PIPELINE FOR BEST MODEL - RANDOM FOREST

```
numeric_features = X_train.describe().columns

# Create a transformer for numeric columns
numeric_transformer = Pipeline(
    steps=[
        # missing values --> by default mean
        ('scaler', StandardScaler())
    ]
)


Random_regression = Pipeline(
    steps=[
        ('Randomregression', RandomForestClassifier(n_estimators = 250, max_depth = 50, random_state = 42))
    ]
)
```

```
# Fit the model
Random_regression.fit(X_train, y_train)
```

MODELS SUMMARY

We have tried several classification models to come up with the best model that can predict the change category depending on the high price, low price, open price, and close price.

Then we select the Random forest as the best model based on the highest weighted avg of f1.

RESULTS

Future Work

Thank You For Listening

ANY QUESTION?

TEAM MEMBERS :

- Maha
- Fatima
- Rasha
- Joharah
- Areej
- Samar