Istanbul Stock Exchange Linear Regression and LSTM Prediction

Author: Anthony Leary, an134404, Optional Project

***Abstract*—This research paper conducts a linear regression test to try to create a predictive model for the Standard & Poor's (SAP) 500 Return Index regarding working days in Istanbul Stock Exchange. After finding the predictive relationship between SAP 500 and the other international stock market indices, a linear regression predictive model is created to find a predictive relationship between the Istanbul Stock Exchange and the other stock market indices.**

I. Introduction

This dataset from "datasciencedojo.com" includes 536 rows and 9 columns of returns of the Istanbul Stock Exchange with seven other international indices: SP, DAX, FTSE, NIKKEI, BOVESPA. MSCE_EU, and MSCI_EM from June 5, 2009, to February 22, 2011, and it is organized regarding working days in the Istanbul Stock Exchange. This dataset can be used to develop skills in exploratory data analysis, data visualization, regression, and classification modeling techniques. In these tests, regression modeling techniques will be used, where the performance metrics will be taken from. As a growing student in AI and data science, I hope to gain valuable skills and learn from going through training this dataset to create predictions. The main tasks in this study are to get the performance metrics from two linear regression models: SAP500 and the ISE.

Data Dictionary

Column Position	Atrribute Name	Definition	Data Type	Example	% Null Ratios
1	Date	Date	Quantitative	13-Jan-09, 23-Jan-09, 30-Jan-09	0
2	ISE	Stock Exchange Returns	Quantitative	0.025426, -0.022692, 0.046831	0
3	ISE.1	Istanbul Stock Exchange - National 100 Index	Quantitative	0.031813, -0.044349, 0.061708	0
4	SP	Standard & Poor's 500 Return Index	Quantitative	0.007787, -0.054262, 0.005538	0
5	DAX	Stock Market Return Index of Germany	Quantitative	0.008455, -0.01155, 0.034787	0
6	FTSE	Stock Market Return Index Of UK	Quantitative	0.012866, -0.009351, 0.037891	0
7	NIKKEI	Stock Market Return Index Of Japan	Quantitative	0.004162, 0.003239, -0.008182	0
8	BOVESPA	Stock Market Return Index Of Brazil	Quantitative	0.01892, -0.013151, 0.009838	0
9	EU	MSCI European Index	Quantitative	0.011341, -0.012045, 0.0328	0
10	EM	MSCI Emerging Markets Index	Quantitative	0.008773, -0.004029, 0.01032	0

II. Tasks

A. Create a predictive model for SAP 500 using linear regression techniques

B. Create a predictive model for ISE using linear regression techniques

C. (Stretch) Create a LSTM model for a time seres predictive model of SAP 500.

III. Definitions

A. Linear Regression – used as an algorithm to predict a relationship between two different features / variables. The two kinds are the independent and dependent variables. The test set leads itself to be the dependent variable. In the linear regression tasks running for these predicive models, each obervation is studied.

B. Long short-term memory (LSTM) – this advanced recurrent neural network, which is a sequential netowrk, allows information to continue and can be good for time series analysis. It also solves the RNN's problem of having the vanishing gradient.

C. Pandas – Python libarary used for data manipulation and analysis.

D. Numpy – Python library used for working with arrays. Important for creating proper sizes of data inputs for modeling tasks.

E. Matplotlib – Visualization library for Python, built on numpy arrays.

F. Plotly – Python library that can be used for data visualization, and gives more information that matplotlib. Supports various types of plots.

G. Scikit-learn (Sklearn) – Useful and robust library for machine learning in Python. The LinearRegresson model in sklearn will be used for creating our model predictions.

IV. Preprocess Data

A. Load in dataset

- Using the pandas library we can load in the dataset from my GitHub repository using the "pd.read_csv(url)" function.

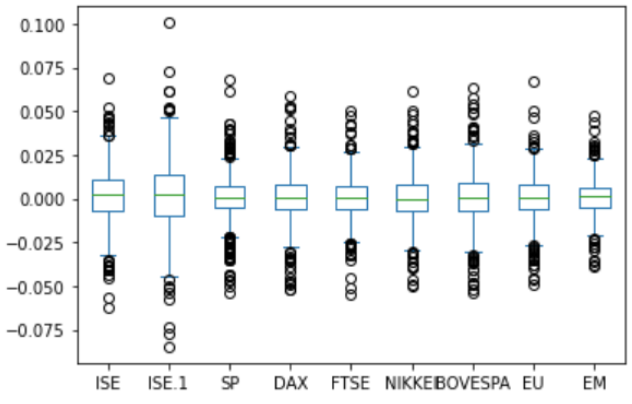
- It is important to load in the data set from a public domain so that it can be accessed and ran by anyone during the testing of the model.

After reading in this data with pandas, using another pandas function “file.info()” will display information of the csv file read in including the number of column, column name, length, null, count, and its type.

```
RangeIndex: 536 entries, 0 to 535
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   date      536 non-null    object
 1   ISE       536 non-null    float64
 2   ISE.1     536 non-null    float64
 3   SP        536 non-null    float64
 4   DAX       536 non-null    float64
 5   FTSE      536 non-null    float64
 6   NIKKEI    536 non-null    float64
 7   BOVESPA   536 non-null    float64
 8   EU        536 non-null    float64
 9   EM        536 non-null    float64
dtypes: float64(9), object(1)
memory usage: 42.0+ KB
```

This is important information we can use to determine what columns, or variables, we want to use in our regression tests, make sure there are the same number of data points in each column, and the type is important to know how the tester can manipulate it.

B. Further analyze the data

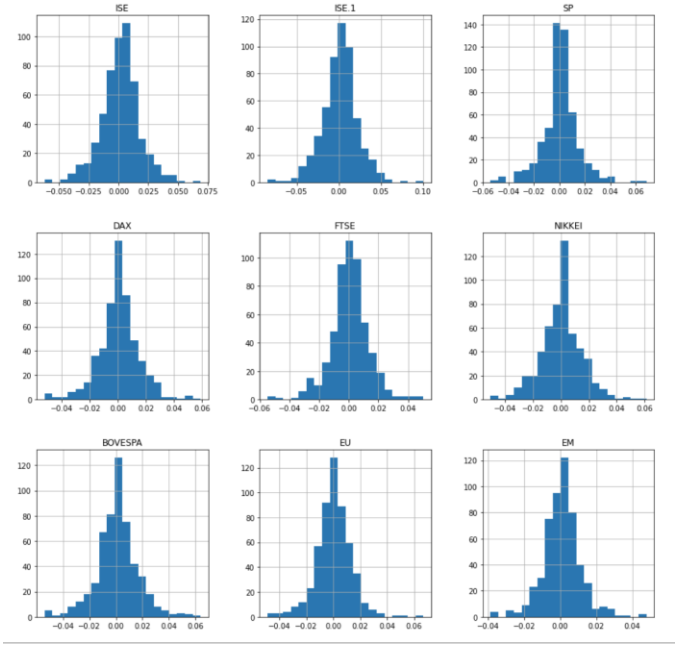


Making a box plot can help us check for any outliers in the data. Outliers are any data points that are far away from the mean, with no other data points nearby. Here, we can see there is a small outlier in the “ISE.1” column. Although, it is an outlier, it does not appear to have a drastic affect on the majority of the data given.

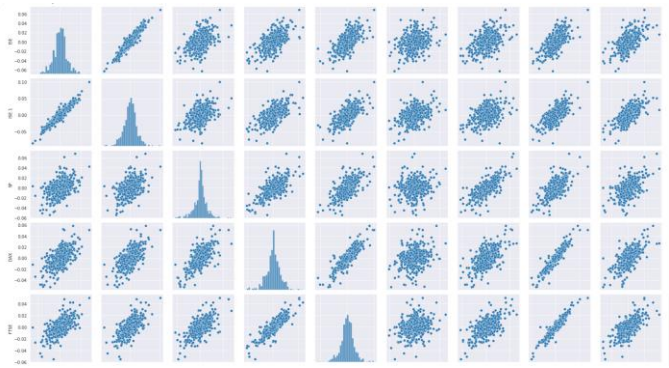
We can use another pandas function, “.describe()”, to print the important statistical information of the dataset.

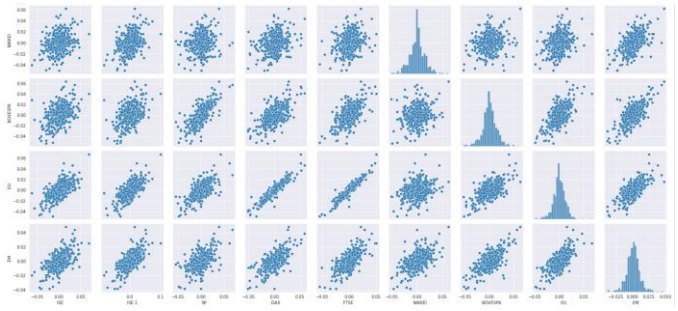
	ISE	ISE.1	SP	DAX	FTSE	NIKKEI	BOVESPA	EU	EM
count	536.000000	536.000000	536.000000	536.000000	536.000000	536.000000	536.000000	536.000000	536.000000
mean	0.001629	0.001552	0.000643	0.000721	0.000510	0.000308	0.000935	0.000471	0.000936
std	0.016264	0.021122	0.014093	0.014557	0.012656	0.014850	0.015751	0.012990	0.010501
min	-0.062208	-0.084716	-0.054262	-0.052331	-0.054816	-0.050448	-0.053849	-0.048817	-0.038564
25%	-0.006669	-0.009753	-0.004675	-0.006212	-0.005808	-0.007407	-0.007215	-0.005952	-0.004911
50%	0.002189	0.002643	0.000876	0.000887	0.000409	0.000000	0.000279	0.000196	0.001077
75%	0.010584	0.013809	0.006706	0.008224	0.007428	0.007882	0.008881	0.007792	0.006423
max	0.068952	0.100621	0.068366	0.058951	0.050323	0.061229	0.063792	0.067042	0.047805

We can see the overall mean values are very close to 0 for each return index. This can lead us to believe that there could be white noise time series to be considered. White noise in a time series can lead a tester to know that a valuable prediction cannot be made across the set.



Visualizing this data can show us the overall analysis of each of the statistical models for each exchange.





We can further visualize our data by using another python library, seaborn, to display a model comparing each set of data to the other to look for a relationship. This is called exploratory data analysis, which can then use to find a good linear model. Using the plots above, we can see there are some linear relationships between some of the exchanges.

C. *Create Training, Validation, and Test data frames*

To create our training, validation, and test data frames, we will continue to use the panda's library functions from the data object we read our dataset into. For this linear regression model, a regular 70-15-15 split is used for the training, testing, and validation data frames respectively.

```
# Training Split
training_nums = len(records) * .70
training_nums = math.ceil(training_nums)
training_data = records.iloc[:training_nums, :]

# Testing Split
testing_nums = len(records) * .15
testing_nums = math.ceil(testing_nums)
testing_data = records.iloc[training_nums:(testing_nums+training_nums), :

# Validation Split
validation_nums = len(records) * .15
validation_nums = math.ceil(validation_nums)
validation_data = records.iloc[(testing_nums+training_nums): ]

X_train = training_data.iloc[:, [1,2,4,5,6,7,8]].values
y_train = training_data.iloc[:, 3]

X_test = testing_data.iloc[:,[1,2,4,5,6,7,8]].values
y_test = testing_data.iloc[:, 3].values

X_val = testing_data.iloc[:,[1,2,4,5,6,7,8]].values
y_val = testing_data.iloc[:, 3].values
```

V. Balance Training Data Set

Since there is no stepwise optimization process in a linear regression model, feature scaling is not necessary for this test. If we were using a LSTM model to make our time series prediction, we would need to use MinMax scaling, or some other scaling algorithm to properly balance the training data set before it can be used for testing.

VI. Build The Predictive Model

This model will be built using sklearn's LinearRegression() method. Sklean's built in models make it simple to use, and to not have to program a linear regression model from scratch.

```
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)
y = linear_model.predict(X_train)
```

A. *Visualize the Actual Vs Predictive Data*

In this testing, we can use matplotlib to visualize our data. This will give us a graph, where we can differentiate the data sets by color.

```
# Visulize the training data compared to the actual return price
plt.plot(y_test, color='red', label='Actual Return Price')
plt.plot(linear_model.predict(X_test), color='green', label = 'Predicted Return Price')
plt.title('SAP Stock Prediction')
plt.xlabel('Time')
plt.ylabel('Stock Price')
plt.legend()
plt.show()
```

VII. Calculate Performance Metrics

Some important metrics that are considered in a linear regression model include, its Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and its R2 score.

Mean square error tells us the difference between the actual and estimated values for our prediction. A score close to 0, tells the modeler their model is very close to the actual values; hence it would be a good model.

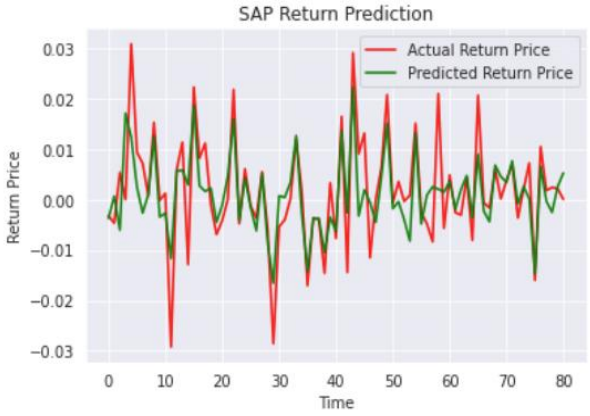
R2 in linear regression represents the proportion of the variance for a dependent variable that is explained by the independent variable. Hence, it tells us how well data will fit the regression model that is created.

Formatting our visualization output nicely, we can use the sklearn library functions to make predictions on our training, testing, and validation sets, which will give us the linear regression performance metrics of MSE and r^2.

```
scores = f'''
{'Metric'.ljust(10)}{'Train'.center(20)}{'Test'.center(20)}{'Validation'.center(20)}
{'r2_score'.ljust(10)}{r2_score(y_train, linear_model.predict(X_train))}\t{r2_score(y_test, linear_model.predict(X_test))}\t{r2_score(y_val, linear_model.predict(X_val))}
{'MSE'.ljust(10)}{mse(y_train, linear_model.predict(X_train))}\t{mse(y_test, linear_model.predict(X_test))}\t{mse(y_val, linear_model.predict(X_val))}
'''
```

Running the pandas Data Frame function, we can create a table to side by side look at the true data values, vs our predicted values.

Metric	Train	Test	Validation
r2_score	0.6421502322727681	0.5989675005048614	0.3266355197052253
MSE	8.849437614031473e-05	4.662889684244367e-05	3.130134406534611e-05



	True Value	Predicted Value
0	-0.003246	-0.003745
1	-0.004673	0.000641
2	0.005345	-0.006083
3	0.000000	0.017073
4	0.030850	0.012304
...	...	...
76	0.010469	0.006527
77	0.001772	-0.000303
78	0.002386	-0.002548
79	0.002145	0.002569
80	0.000017	0.005264

A. *Review the prediction visualization results*

With a review of our data frames, it is shown that a r2 score of ~0.64 for the training set, ~0.60 for the testing set, and ~0.33 for the validation set, this predictive model is decent, but far from being considered predictable.

Looking at the comparisons of all the models against each other in the seaborn visual, I thought of a claim to use linear regression to make a predictive model of what the ISE returns will be. It is shown to have linearity with the data, compared to the data of SAP 500.

VIII. Imrprove the Predictive Model

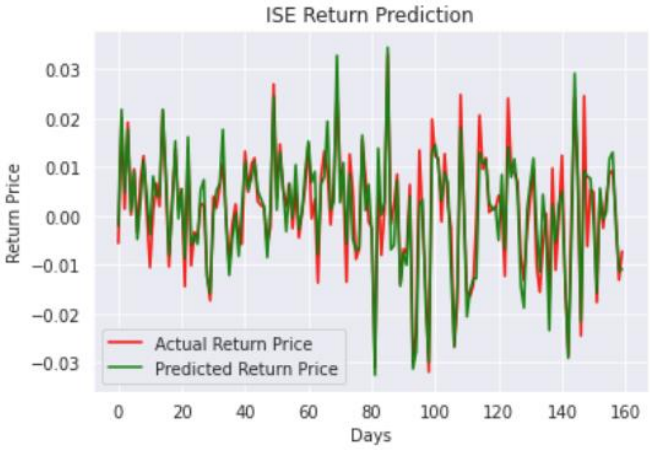
As stated, to improve on finding a predictive model for this dataset, a test will be made this time to find a predictive model for the ISE.

After slicing the data properly for preprocessing of this model, the steps are replicated as before to achieve visualizations of the models performance metrics.

A. *Visualizations of the ISE linear regression predictive model metrics*

Metric	Train	Test
r2_score	0.920974092203519	0.8440510277899669
MSE	2.4958757742446816e-05	2.2113186183923273e-05

With an R2 score of .92 in the training data frame, and .84 in the testing data frame, we can visually see this is a good predictive model for the ISE against the data.



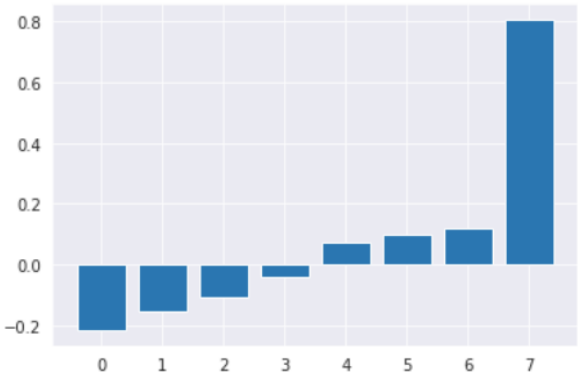
	True Value	Predicted Value
0	-0.005581	-0.002025
1	0.020009	0.021717
2	0.001420	0.004838
3	0.019062	0.017665
4	0.000190	0.000923
...	...	...
155	0.008599	0.011618
156	0.009310	0.012988
157	0.000191	-0.001845
158	-0.013069	-0.011231
159	-0.007246	-0.010886

Reviewing the chart differentiating the actual return prices vs the predicted return prices and looking at the true value vs the predicted value in the data frame chart, we can see this is an accurate predictive model.

IX. Feature Importance

Using the 'coeff_' property of sklearns LinearRegression model, it contains the coefficient found for all the input variables, and they can provide information to the overall feature importance score.

```
Feature: 0, Score: -0.21577
Feature: 1, Score: -0.15259
Feature: 2, Score: -0.10554
Feature: 3, Score: -0.03928
Feature: 4, Score: 0.07333
Feature: 5, Score: 0.09679
Feature: 6, Score: 0.11815
Feature: 7, Score: 0.80536
```



References

[1] https://realpython.com/linear-regression-in-python/

[2] https://www.askpython.com/python/built-in-methods/python-iloc-function

[3] https://inria.github.io/scikit-learn-mooc/python_scripts/dev_features_importance.html

[4] https://www.statology.org/simple-linear-regression-in-python/

[5] https://www.shanelynn.ie/pandas-iloc-loc-select-rows-and-columns-dataframe/

[6] https://machinelearningmastery.com/white-noise-time-series-python/#:~:text=White%20noise%20is%20an%20important%20concept%20in%20time,improvements%20could%20be%20made%20to%20the%20predictive%20model

[7] https://www.unite.ai/what-is-linear-regression/

[8] https://www.mygreatlearning.com/blog/mean-square-error-explained/

[9] https://www.wallstreetmojo.com/r-squared-formula/