# Overview

## Objective:

Obtain a preliminary model of Disney's stock data to predict the closing price based on a full and reduced model of the data. Some of the techniques used include simple linear regression, feature reduction and importance, as well as a comparison of the Mean Square Error.

## Data:

The data in question centers around Disney stock data ranging back from January 1962 to August 2024. We will not currently focus on time; therefore, the dates will be removed from the list of predictor variables. Predictors within the dataset that will influence the model includes the opening price, the highest value of that day, the lowest value, and the volume of stock sold.

## Data Source:

Patel, K. (2024, August 28). Disney Stock Data. Kaggle. https://www.kaggle.com/datasets/krupalpatel07/disney-stock-data

In [55]:

```
import pandas as pd
import numpy as np

# Load the data
disney_stocks = pd.read_csv(
    filepath_or_buffer = "C:/Users/brink/OneDrive/Desktop/DIS.csv",
    engine = 'pyarrow',
    dtype = {
        'Date': str,
        'Open': float,
        'High': float,
        'Low': float,
        'Close': float,
        'Volume': int
    }
)

# Set a random seed for reproducibility
np.random.seed(823)

#Establish Partitions
list_partition = ['Train','Test']
disney_stocks['partition'] = np.random.choice(
```

```
    a = list_partition,
    size = disney_stocks.shape[0]
)
disney_stocks.head()
```

Out[55]:

| | Date | Open | High | Low | Close | Volume | partition |
|---|---|---|---|---|---|---|---|
| **0** | 1962-01-02 | 0.057941 | 0.059886 | 0.057941 | 0.057941 | 841958 | Train |
| **1** | 1962-01-03 | 0.057941 | 0.058914 | 0.057941 | 0.058719 | 801865 | Test |
| **2** | 1962-01-04 | 0.058719 | 0.058914 | 0.058331 | 0.058719 | 962238 | Train |
| **3** | 1962-01-05 | 0.058719 | 0.059108 | 0.058525 | 0.058914 | 962238 | Test |
| **4** | 1962-01-08 | 0.058914 | 0.059691 | 0.057553 | 0.058719 | 1282984 | Test |

In [57]:

```
#Column Organization
list_categorical = [
    'Date'
]
list_numeric = [
    'Open', 'High','Low', 'Volume'
]
disney_stocks.columns[~disney_stocks.columns.isin(list_categorical + list_numeric)]
```

Out[57]:

```
Index(['Close', 'partition'], dtype='object')
```

In [59]:

```
#Observing the Correlation Between Variables
for j in list_numeric: disney_stocks[j] = pd.to_numeric(
    arg = disney_stocks[j],
    errors = 'coerce'
)
disney_stocks_corr = disney_stocks[list_numeric].corr(
    method = 'spearman'
)
disney_stocks_corr.round(2)
```

Out[59]:

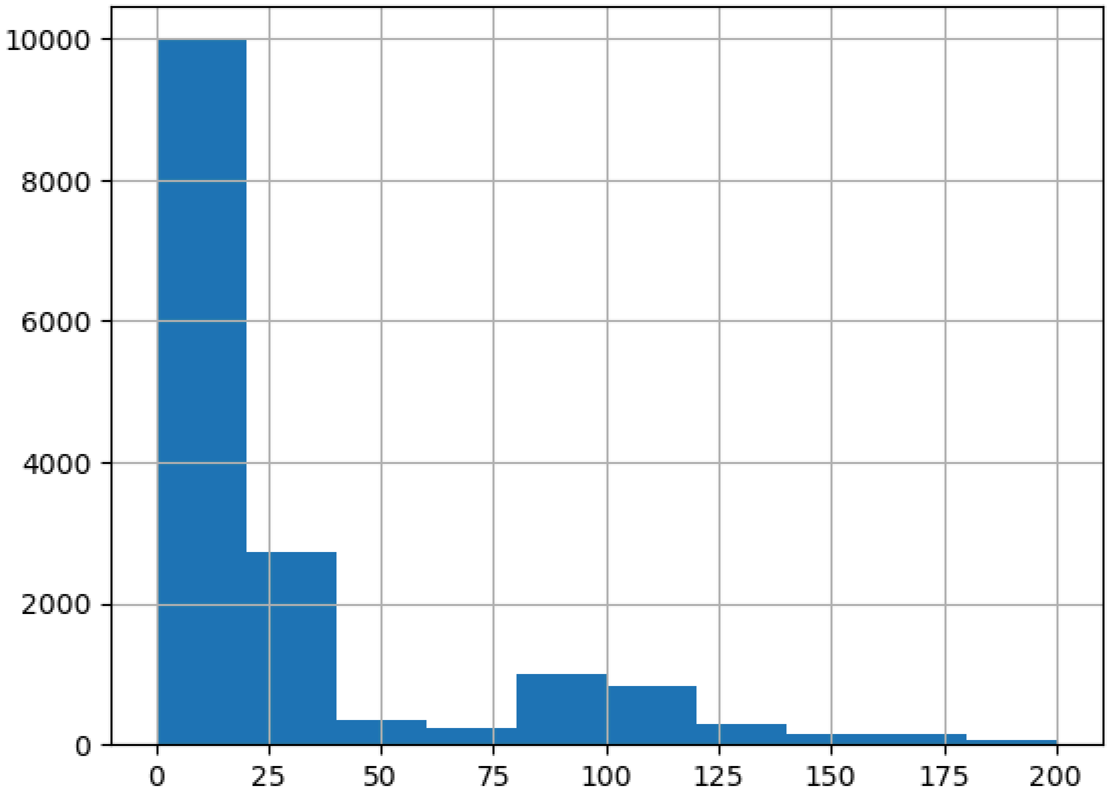
| | Open | High | Low | Volume |
|---|---|---|---|---|
| **Open** | 1.00 | 1.00 | 1.00 | 0.69 |
| **High** | 1.00 | 1.00 | 1.00 | 0.69 |
| **Low** | 1.00 | 1.00 | 1.00 | 0.69 |
| **Volume** | 0.69 | 0.69 | 0.69 | 1.00 |

In [61]:

```
#Visualizing the Target Variable
disney_stocks['Close'].hist()
```

Out[61]:

```
<Axes: >
```




In [63]:
```
#Covert Numerical Data to Numeric
for j in list_numeric: disney_stocks[j] = pd.to_numeric(disney_stocks[j],errors='co
```

In [65]:
```
corr_closing = disney_stocks[['Close'] + list_numeric].corr().abs().sort_values('Cl
list_numeric = corr_closing.index[corr_closing.index != 'Close']
```

In [67]:
```
list_numeric
```

Out[67]:
```
Index(['High', 'Low', 'Open', 'Volume'], dtype='object')
```

In [11]:
```
#Feature Selection
from sklearn.ensemble import RandomForestRegressor
RandomForestRegressor_Closing = RandomForestRegressor().fit(
    X = disney_stocks[list_numeric],
    y = disney_stocks['Close']
)
from sklearn.feature_selection import RFE
RFE_Closing = RFE(
    estimator=RandomForestRegressor_Closing
).fit(
    X = disney_stocks[list_numeric],
    y = disney_stocks['Close']
)
disney_stocks[list_numeric].columns[RFE_Closing.support_]
```

Out[11]:
```
Index(['High', 'Low'], dtype='object')
```

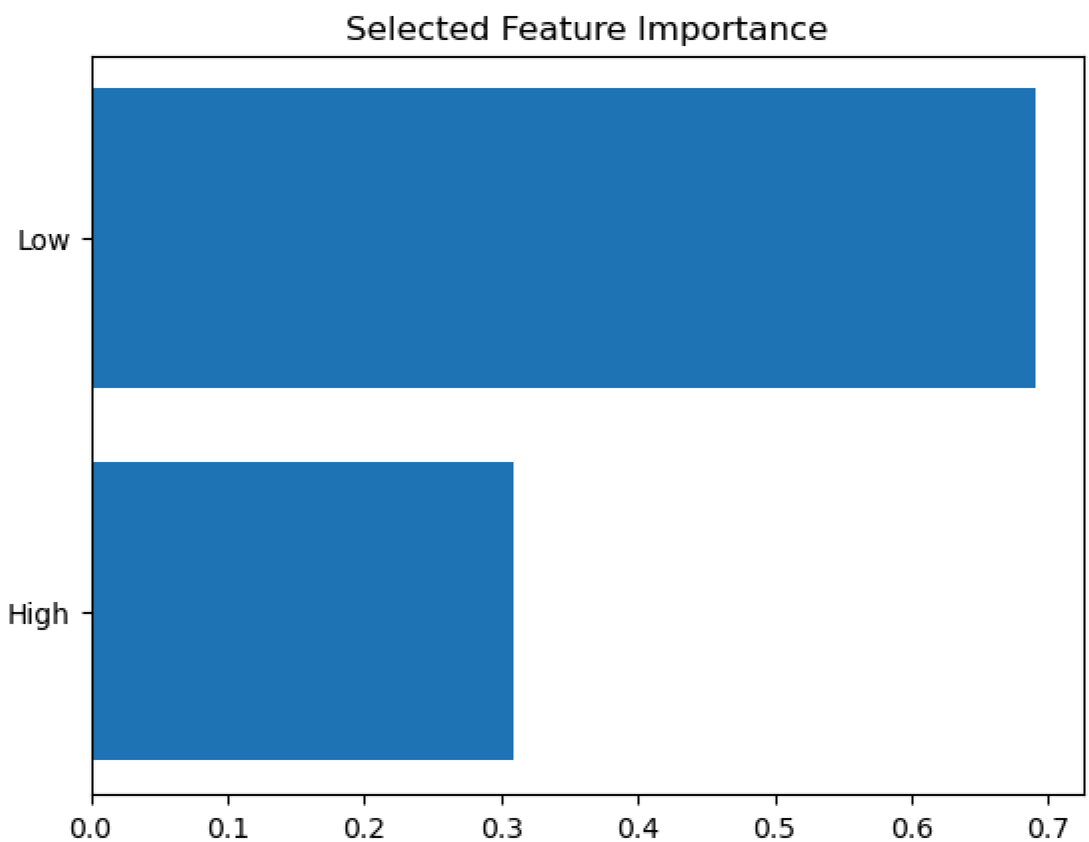
In [12]:
```
#Establish Feature Importance
X=disney_stocks[list_numeric]
```

```
list_full = ['Open', 'High','Low', 'Volume']
list_reduced = ['High', 'Low'
]
RandomForestRegressor_reduced = RandomForestRegressor().fit(
    X = X[list_reduced],
    y = disney_stocks['Close']
)
df_feature_importances_ = pd.DataFrame({
    'feature_importances_' : RandomForestRegressor_reduced.feature_importances_,
    'feature_names_in_' : RandomForestRegressor_reduced.feature_names_in_
}).sort_values('feature_importances_',ascending = False)
df_feature_importances_ = df_feature_importances_.loc[df_feature_importances_.featu
print(df_feature_importances_)
df_feature_importances_.feature_names_in_.tolist()
```

```
   feature_importances_ feature_names_in_
1              0.690477               Low
0              0.309523              High
```

Out[12]: ['Low', 'High']

In [13]:
```
#Visualize the Feature Importance
import matplotlib.pyplot as plt
plt.barh(df_feature_importances_.feature_names_in_,df_feature_importances_.feature_
plt.title('Selected Feature Importance')
plt.gca().invert_yaxis()
plt.show()
```




In [14]:
```
#Establish the Full and Reduced Models for Easier Access in Future Code
dict_predictors = {
```

```
    'full' : list_full,
    'reduced' : list_reduced
}
list_predictors = [
    list_full,list_reduced
]
```

In [15]:
```
#Perform Simple Linear Regression for Each Model
from  sklearn.linear_model import LinearRegression
list_LinearRegression = [
    LinearRegression().fit(
        X = disney_stocks.loc[disney_stocks['partition'] == 'Train'][predictors],
        y = disney_stocks.loc[disney_stocks['partition'] == 'Train']['Close']
    ) for predictors in dict_predictors.values()
]
list_LinearRegression
```

Out[15]:
```
[LinearRegression(), LinearRegression()]
```

In [16]:
```
#Observe Coefficients of the model
list_coef_ = [model.coef_ for model in list_LinearRegression]
list_coef_
```

Out[16]:
```
[array([-5.54548287e-01,  8.01695387e-01,  7.52779588e-01, -1.10038941e-09]),
 array([0.50633845, 0.49362003])]
```

In [17]:
```
[pd.DataFrame({'predictor' : list_predictors[j],'coef_' : list_coef_[j]}).sort_valu
```

Out[17]:
```
[  predictor         coef_
 0      Open -5.545483e-01
 3    Volume -1.100389e-09
 2       Low  7.527796e-01
 1      High  8.016954e-01,
   predictor     coef_
 1       Low  0.493620
 0      High  0.506338]
```

In [18]:
```
#Ranks of the Predictor Variables
[model.rank_ for model in list_LinearRegression]
```

Out[18]:
```
[4, 2]
```

In [19]:
```
#Predictions
list_predict = [
    list_LinearRegression[j].predict(
        X = disney_stocks[list_predictors[j]]
    ) for j in range(len(list_predictors))
]
disney_predict = pd.DataFrame(list_predict).transpose()
disney_predict.columns = ['full','reduced']
disney_predict = pd.concat([disney_stocks[['Close','partition']],disney_predict],ax
disney_predict = pd.melt(
    frame = disney_predict,
    id_vars=['Close','partition'],
    value_vars=['full','reduced'],
```

```
    var_name='model',
    value_name='predict'
)
disney_predict['residual'] = disney_predict['predict'] - disney_predict['Close']
disney_predict['squared_residual'] = disney_predict['residual']**2
disney_predict
```

Out[19]:

| | Close | partition | model | predict | residual | squared_residual |
|---|---|---|---|---|---|---|
| **0** | 0.057941 | Train | full | 0.065797 | 0.007856 | 0.000062 |
| **1** | 0.058719 | Test | full | 0.065062 | 0.006343 | 0.000040 |
| **2** | 0.058719 | Train | full | 0.064747 | 0.006028 | 0.000036 |
| **3** | 0.058914 | Test | full | 0.065050 | 0.006136 | 0.000038 |
| **4** | 0.058719 | Test | full | 0.064324 | 0.005605 | 0.000031 |
| **...** | ... | ... | ... | ... | ... | ... |
| **31527** | 86.300003 | Train | reduced | 86.057141 | -0.242862 | 0.058982 |
| **31528** | 88.790001 | Train | reduced | 88.081764 | -0.708237 | 0.501600 |
| **31529** | 89.300003 | Train | reduced | 89.014242 | -0.285761 | 0.081659 |
| **31530** | 90.820000 | Train | reduced | 90.190532 | -0.629468 | 0.396230 |
| **31531** | 89.739998 | Train | reduced | 89.999393 | 0.259395 | 0.067286 |

31532 rows × 6 columns

In [20]:

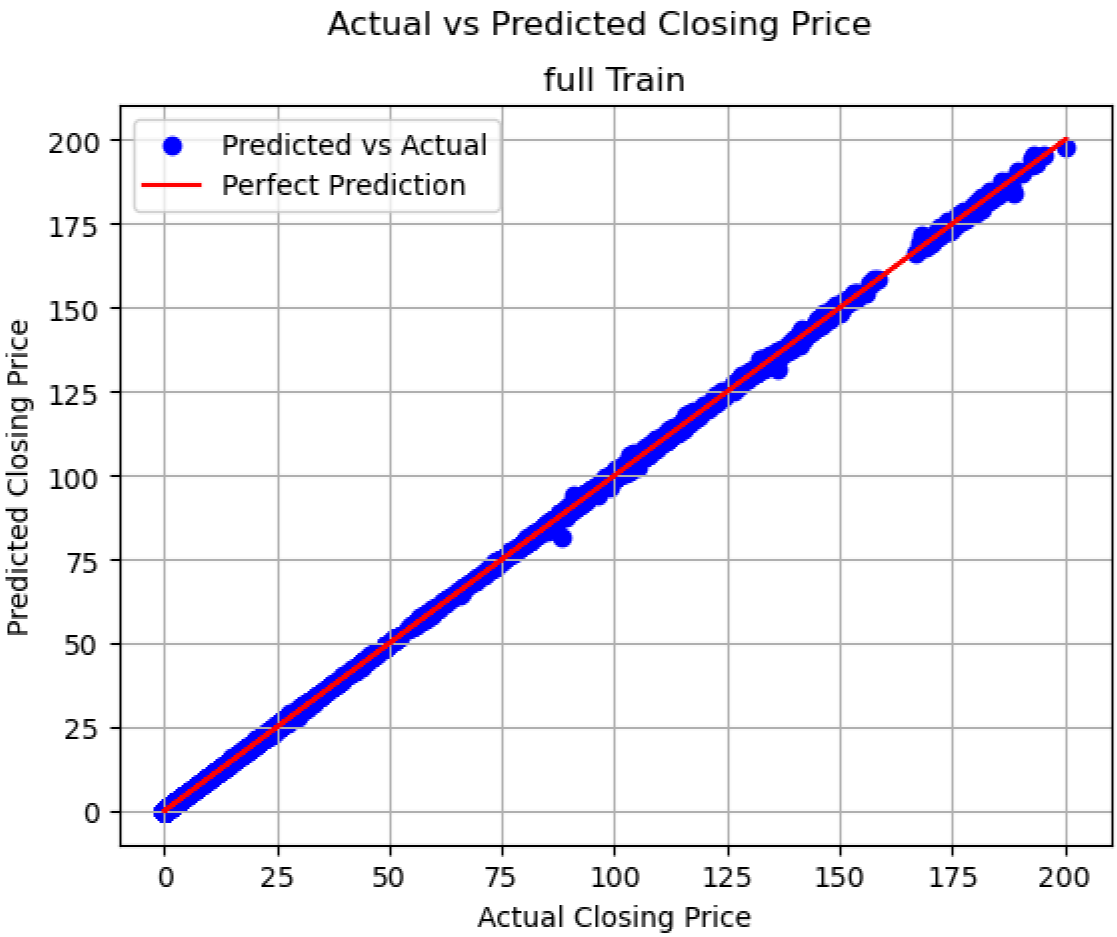
```
#Mean Square Error of Models
disney_predict.groupby(['model','partition'])['squared_residual'].mean()
```

Out[20]:

```
model    partition
full     Test         0.092971
         Train        0.087287
reduced  Test         0.135023
         Train        0.128617
Name: squared_residual, dtype: float64
```

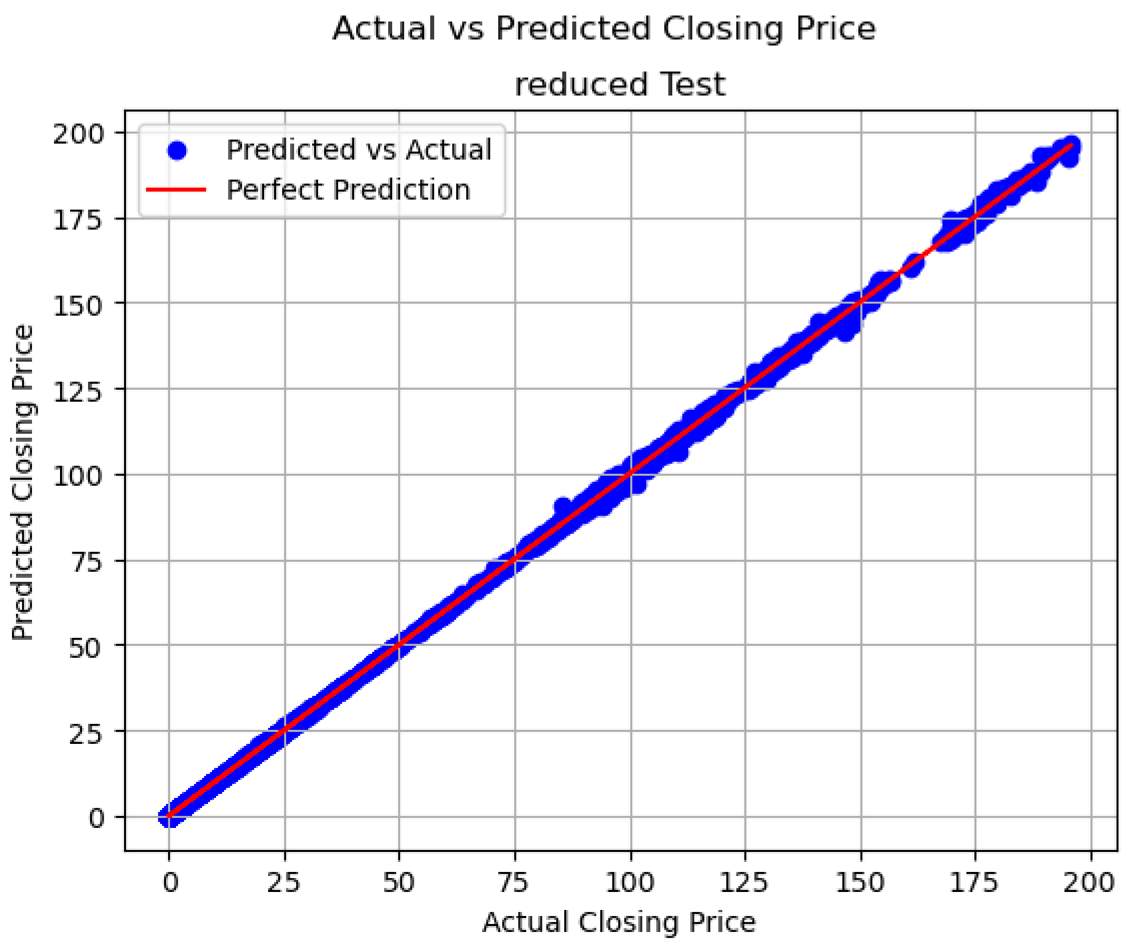
In [21]:

```
#Test Linear Assumptions
import matplotlib.pyplot as plt
for model in ['full','reduced']:
    for partition in ['Train','Test']:
        index = (disney_predict['model'] == model) & (disney_predict['partition'] =
        plt.scatter(disney_predict.loc[index]['Close'], disney_predict.loc[index]['
        plt.plot(disney_predict.loc[index]['Close'], disney_predict.loc[index]['Clo
        plt.suptitle('Actual vs Predicted Closing Price')
        plt.title(model + ' ' + partition)
        plt.xlabel('Actual Closing Price')
        plt.ylabel('Predicted Closing Price')
        plt.legend()
        plt.grid(True)
        plt.show()
```


Actual vs Predicted Closing Price
full Train
Predicted vs Actual
Perfect Prediction
Predicted Closing Price
Actual Closing Price
0
25
50
75
100
125
150
175
200

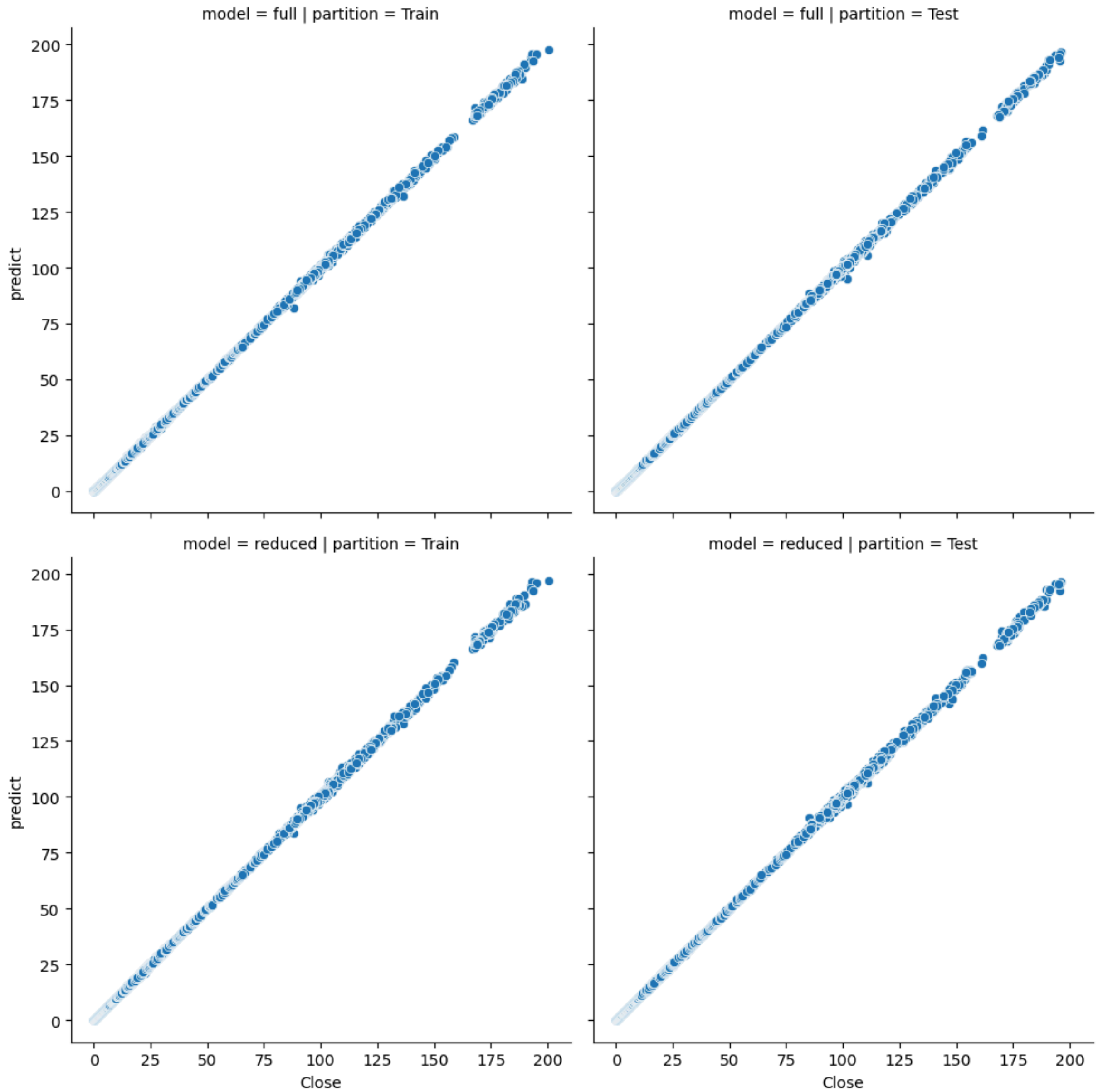

Actual vs Predicted Closing Price
full Test
Predicted vs Actual
Perfect Prediction
Predicted Closing Price
Actual Closing Price
0
25
50
75
100
125
150
175
200


Actual vs Predicted Closing Price
reduced Train
Predicted vs Actual
Perfect Prediction
Predicted Closing Price
Actual Closing Price
0
25
50
75
100
125
150
175
200




In [22]:
```
import seaborn as sns
sns.relplot(
    data=disney_predict.loc[~disney_predict['partition'].isin(['Score'])],
    x='Close',
    y='predict',
    row='model',
    col='partition',
)
```

Out[22]: <seaborn.axisgrid.FacetGrid at 0x202be847140>




In [28]:
```
#Compare MSE
list_models = [RandomForestRegressor_Closing,RandomForestRegressor_reduced]
disney_se = pd.concat([(model.predict(
    X = disney_stocks[model.feature_names_in_]
) - disney_stocks['Close'])**2 for model in list_models],axis = 1)
disney_se.columns = ['full','reduced']
disney_se['partition'] = disney_stocks['partition']
disney_se.groupby('partition').mean().loc[['Train','Test']]
```

Out[28]:

| partition | full | reduced |
|---|---|---|
| **Train** | 0.019896 | 0.024316 |
| **Test** | 0.020881 | 0.025810 |

The model does appear to show a linear relation as the predicted values seemed to match the actual values. Furthermore, the mean square error was not a signifigantly better for the reduced model to make it a reasonable choice (considering the full model has only a few more predictor variables). Thus, the full model would prove most useful for further analysis. The results may be further improved using scaling methods to create a more normal distribution. In addition, the target variable (closing prices) was highly right-skewed. Therefore, further testing will focus on this aspect as well as testing decision trees along with random forest.