max chen

financial modeling

a csci1070 analysis

df cleaning steps

dataset from kaggle

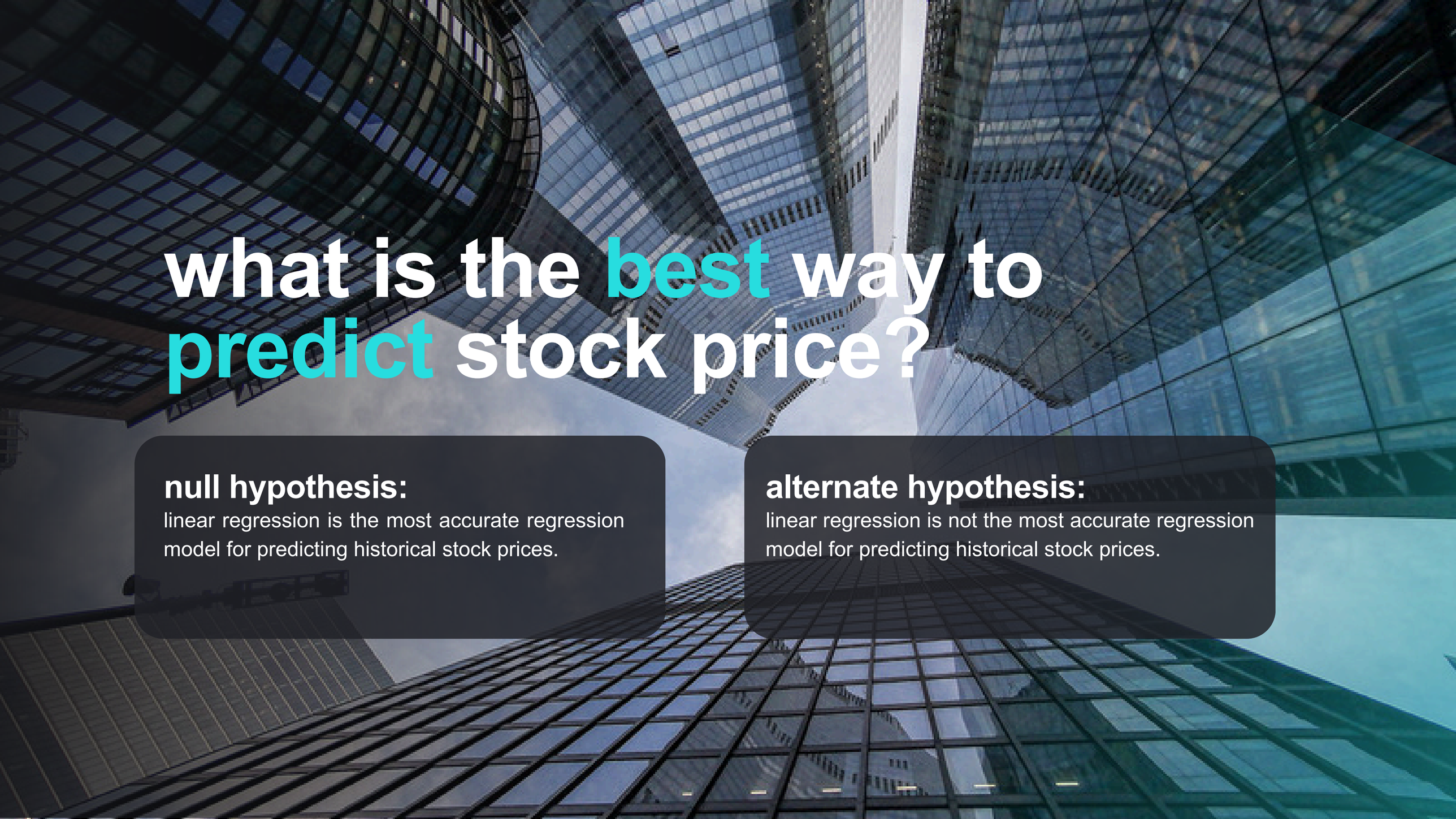
remove dividend and weird data from df

graph data and boxplot to check for outliers

winsorize columns that have outliers

standardize the data for better analysis

*for more detailed cleaning, check cleaning.ipynb



what is the best way to predict stock price?

null hypothesis:
linear regression is the most accurate regression model for predicting historical stock prices.

alternate hypothesis:
linear regression is not the most accurate regression model for predicting historical stock prices.

linear regression

```python
from sklearn import linear_model
from sklearn.model_selection import train_test_split

y = clean_df["price"]

X = clean_df[["liabilities","equity","total_assets","current_assets","total_revenue","net_income","shares_outstanding"]]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=32)

lrm = linear_model.LinearRegression()

lrm.fit(X_train, y_train)
```

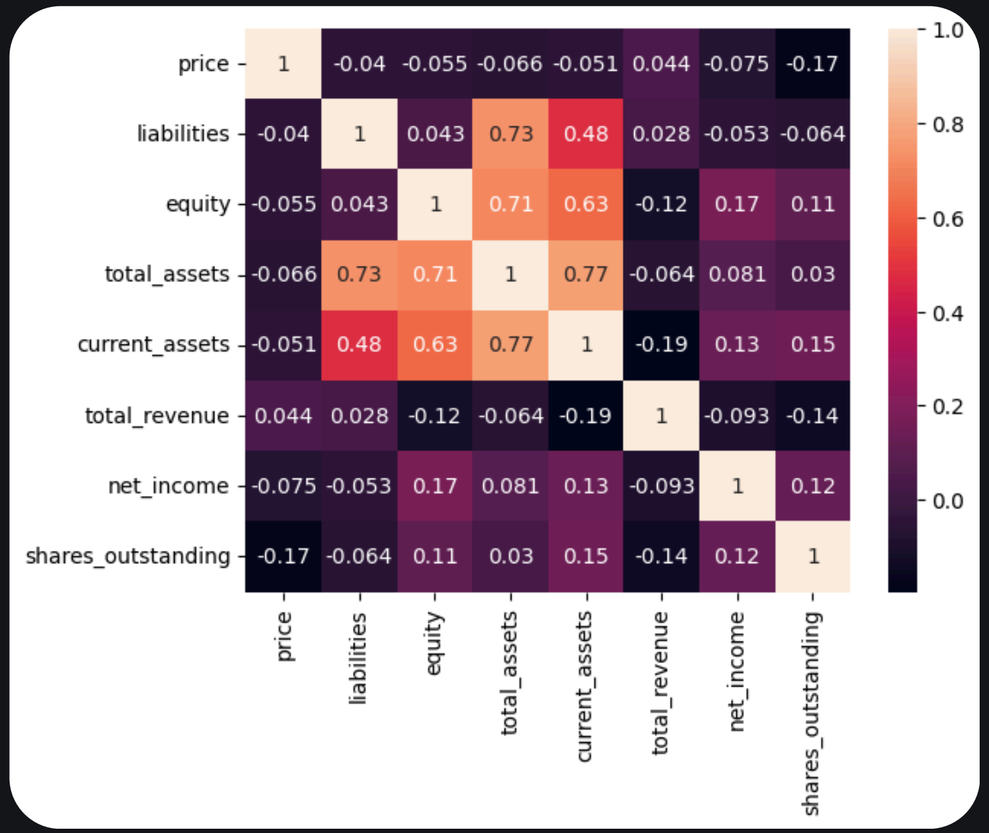
```
LinearRegression
LinearRegression()
```

accuracy?

first things first, I implemented linear regression.
but it didn't do so well... but what else can we try?

```python
accuracy_score = lrm.score(X_test, y_test)
accuracy_score
```

```
-0.14398823805374228
```



confusion matrix

?

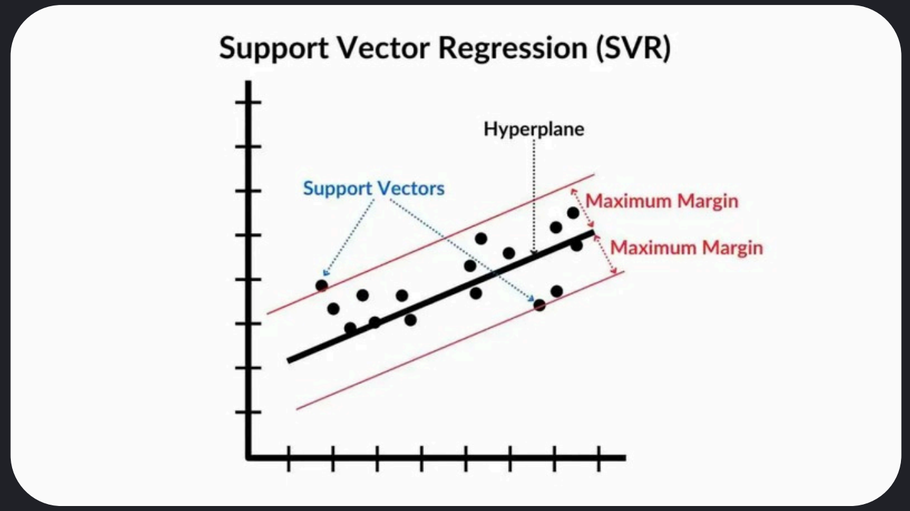
why didn't it fit?

from the confusion matrix, it's clear that price doesn't strongly correlate with any variable, so it makes sense that linear regression is weak.

what’s the solution?

Support Vector Regression

after some research, I found another more well-established regression known as Support Vector Regression (SVR).
SVR is quite complicated, but what you need to know is that is better for **modeling high-dimensional data**, or data with lots of variables, just like stock data.



- uses a hyperplane to classify data
- has support vectors to help position hyperplane
- margin of error to fit less outliers

implementing SVR

for the first implementation of SVR it resulted in an accuracy score of **0.44**, not bad but could be better.

```
svr.score(X,y)
```

```
0.4402405953517664
```

in order to fine-tune the SVR, I played around with the **hyperparameters** until I got a higher accuracy.

```
svr1 = SVR(kernel='rbf',gamma=0.5, C=20, epsilon = 0.02)
```

```
0.7051278624676703
```

what do these mean?

gamma - controls weight of each data point

epsilon - how much error it will tolerate

C - prioritize strict classification

one more step!

time series, differencing

we can help our SVR just a bit more, through differencing! but first: what are time series?

time series -

data points collected at regular time intervals, representing how a variable changes over time (e.g., daily stock prices, monthly sales, or annual temperatures).

differencing -

similar to taking the first derivative of a function, take two points, find the difference, and replace. you will get one less data point from this.

```
clean_df["total_revenue"] = clean_df["total_revenue"].diff()
clean_df["liabilities"] = clean_df["liabilities"].diff()
clean_df["equity"] = clean_df["equity"].diff()
clean_df["total_assets"] = clean_df["total_revenue"].diff()
clean_df["current_assets"] = clean_df["current_assets"].diff()
clean_df["current_liabilities"] = clean_df["current_liabilities"].diff()
clean_df["net_income"] = clean_df["net_income"].diff()
clean_df["shares_outstanding"] = clean_df["shares_outstanding"].diff()
```

```
clean_df = clean_df.dropna()
```

this helps a lot with time series data, as it helps to remove two things: **trends and seasonality.** doing this will help the SVR ignore other patterns within the data, and help it focus on the variables we want analyzed

finale

after all that, our SVR score is damn near perfect!

```
svr1.score(X,y)
```

```
0.9995955160783419
```

what did we learn from this?

for starters, linear regression is NOT the best way to predict stock prices. this is because of the many factors that have to be considered when determining stock price.

but what we did find out is that SVR is a much better model for stocks. after a bit of fine-tuning with the hyperparameters, and tweaking the data through differencing, we were able to create a model with **99.96%** accuracy!

thank you