# Part 1: S&P 500 Stock Price Forecast

> **Motivation:** Beginning in March 2020 during the COVID pandemic, the stock market caught an unexpected upwind. I want to know if it would have been possible to predict this trend early on.

In [123]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import plotly.express as px
import datetime
import yfinance as yf
from scipy import stats

from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression
import keras.backend as K
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dropout
from keras.layers import Dense
from keras.callbacks import EarlyStopping
```

## I) EDA

In [124]:
```
def candlestick(df, title, begin_year='1975'):
    # Plot candlestick chart
    temp = df[df.index > begin_year]
    fig = go.Figure(data=[go.Candlestick(x=temp.reset_index()['Date'],
                open=temp['Open'],
                high=temp['High'],
                low=temp['Low'],
                close=temp['Adj Close'])])

    fig.update_layout(title=title,
                      yaxis_title="USD")

    fig.show()
```

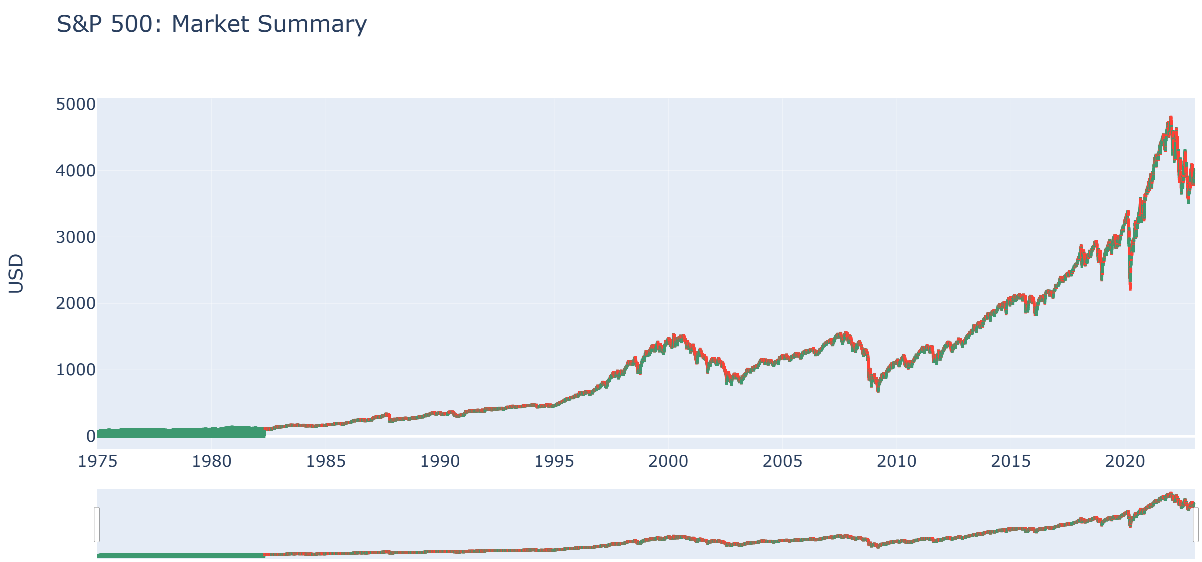
### Analyze the Stock Market

The S&P 500 Index is largely considered an essential benchmark of the stock market. I will analyze it to gain insight about the market

In [125]:
```
SP500 = yf.download('^GSPC')
```

```
[*********************100%***********************]  1 of 1 completed
```

In [126]:
```
# Generic plot of S&P 500 stock price
candlestick(SP500,'S&P 500: Market Summary')
```




In [127]:
```
def annual_percent_change(df):
    # Calculates percent annual return
    # Input: df[Date, Open, High, Low, Close, Adj Close]
    # Output: df[Year, Open, High, Low, Close, % Change]]

    years = np.unique(np.array(SP500.index.year))[0:-1]
    percent_change = []
    year_open, year_close = [], []
    year_low, year_high = [], []

    for i in years:
        temp = df[(df.index >= str(i)) & (df.index < str(i+1))]['Adj Close']
        percent_change.append(round((temp[-1] / temp[0] - 1) * 100, 2))
        year_open.append(temp[0])
        year_close.append(temp[-1])
        year_low.append(min(temp))
        year_high.append(max(temp))

    result = pd.DataFrame({'Date':years,
                          'Open': year_open,
                          'Close': year_close,
                          'Low': year_low,
                          'High': year_high,
                          '% Change': percent_change}).set_index('Date')
    return result

SP500_YoY = annual_percent_change(SP500)
```

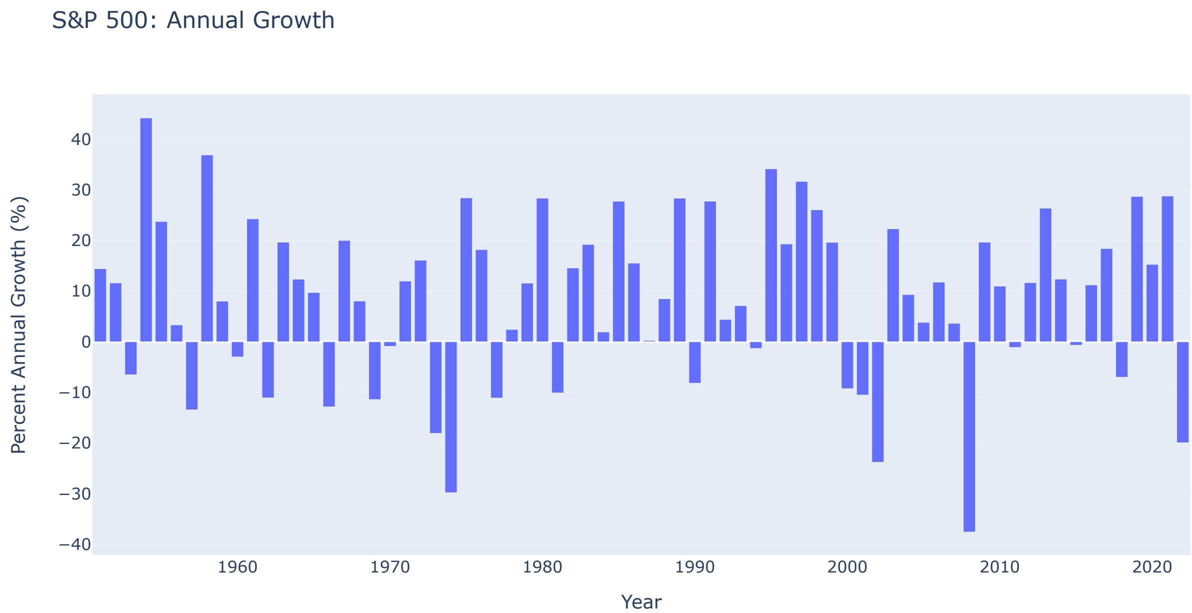
In [128]:
```
average_annual_return = np.mean(SP500_YoY[SP500_YoY.index >= 1950]['% Change'])

SP500_2019 = SP500_YoY[SP500_YoY.index == 2019]['% Change']
SP500_2020 = SP500_YoY[SP500_YoY.index == 2020]['% Change']
SP500_2021 = SP500_YoY[SP500_YoY.index == 2021]['% Change']

SP500_2019, SP500_2020, SP500_2021, average_annual_return
```

Out[128]:
```
(Date
 2019    28.71
 Name: % Change, dtype: float64,
 Date
 2020    15.29
 Name: % Change, dtype: float64,
 Date
 2021    28.79
 Name: % Change, dtype: float64,
 8.894246575342464)
```

The S&P500 index has delivered an average annual growth rate of 8.7% since 1950. However, this rule of thumb did not apply during the pandemic.

- In 2019, S&P 500 grew **29%**
- In 2020, S&P 500 grew **15%**
- In 2021, S&P 500 grew **29%**

In [129]:
```
# Plot % Annual Growth
temp = SP500_YoY[SP500_YoY.index > 1950].reset_index() # Only include recent data, (1950 and beyond)
fig = px.bar(temp, x="Date", y="% Change", title='S&P 500: Annual Growth')
fig.update_layout(xaxis_title='Year', yaxis_title='Percent Annual Growth (%)')
```




In [130]:
```
# Calculate percentile
stats.percentileofscore(SP500_YoY['% Change'], SP500_2019), stats.percentileofscore(SP500_YoY['% Change'], SP500_2020), stats.per
```

Out[130]:
```
(array([90.625]), array([64.58333333]), array([91.66666667]))
```

The plot above shows the percent annual growth in price between 1950 and 2022. We occasionally see long positive streches, often followed by a big negative year. The 2019-2022 market is certainly bullish, with a growth rate in the **90th, 60th, and 90th percentile** which is reminiscent of the **dot-com bubble in the late 90s.**

## II) Modeling and Prediction: one day in the future

- **Objective**: Predict stock prices one day in advance.
- Because future stock prices are very reliant on past prices, I will use the LSTM model which is known for storing past information

### Preprocessing

In [153]:
```
def split(df, split_ratio):
    # Takes in a df and splits it into train and test df

    numrows_train = round(split_ratio * df.shape[0])
    train_df = df[0:numrows_train]
    test_df = df[numrows_train:]
    return train_df, test_df
```

In [154]:
```
df = SP500
split_ratio = 0.8

train_df, test_df = split(df, split_ratio)
```

In [155]:
```
def offset(array, n_features):
    # array: an array of normalized training or test data
    # n_features: use past n days of data as the feature to predict price
    # lookahead: how many days in the future you want to predict
        # Output: x_train and y_train
    x, y = [], []

    for i in range(n_features, len(array)-1):
        x.append(array[i-n_features:i])
        y.append(array[i, 0])

    return np.array(x), np.array(y)
```

In [156]:
```
scaler = MinMaxScaler(feature_range = (0,1))
n_features = 30 # Use the last 60 days as a feature

# Normalize training and test data
train_scaled = scaler.fit_transform(train_df['Adj Close'].values.reshape(-1,1))
test_scaled = scaler.fit_transform(test_df['Adj Close'].values.reshape(-1,1))

x_train, y_train = offset(train_scaled, n_features)
x_test, y_test = offset(test_scaled, n_features)
```

### Build model

Future stock prices are very reliant on past prices, so I will use the LSTM model that can <...>

In [157]:
```
def build_model():
    K.clear_session()
    model = Sequential()

    model.add(LSTM(units = 50, return_sequences = True, input_shape = (x_train.shape[1],1)))
    model.add(Dropout(0.2))

    model.add(LSTM(units = 50, return_sequences = True))
    model.add(Dropout(0.2))

    model.add(LSTM(units = 50))
    model.add(Dropout(0.2))

    model.add(Dense(units=1))

    return model
```

In [158]:
```
model = build_model()
model.compile(loss='mean_squared_error', optimizer='adam')
early_stop = EarlyStopping(monitor='loss', patience=10, verbose=1)
```

### Train model

In [159]:
```
model = build_model()
model.compile(loss='mean_squared_error', optimizer='adam')

model.fit(x_train, y_train, epochs=6,
          batch_size = 32)
```

```
Epoch 1/6
597/597 [==============================] - 10s 13ms/step - loss: 0.0013
Epoch 2/6
597/597 [==============================] - 8s 13ms/step - loss: 6.2116e-04
Epoch 3/6
597/597 [==============================] - 8s 13ms/step - loss: 5.6431e-04
Epoch 4/6
597/597 [==============================] - 7s 13ms/step - loss: 5.2829e-04
Epoch 5/6
597/597 [==============================] - 8s 13ms/step - loss: 4.9433e-04
Epoch 6/6
597/597 [==============================] - 8s 13ms/step - loss: 4.6595e-04
```

Out[159]:
```
<keras.callbacks.History at 0x1b61a8eb250>
```

In [160]:
```
## Runtime Optimization, the model did not improve much past 6 epochs
# epoch, loss = [], []

# for i in range(3, 10): # Optimal num epoch
#     model = build_model()
#     model.compile(loss='mean_squared_error', optimizer='adam')
#     early_stop = EarlyStopping(monitor='loss', patience=10, verbose=1)

#     model.fit(x_train, y_train, epochs=i,
#   batch_size = 32)

#     epoch.append(i)
#     loss.append(model.evaluate(x_test, y_test))
```

### Predict

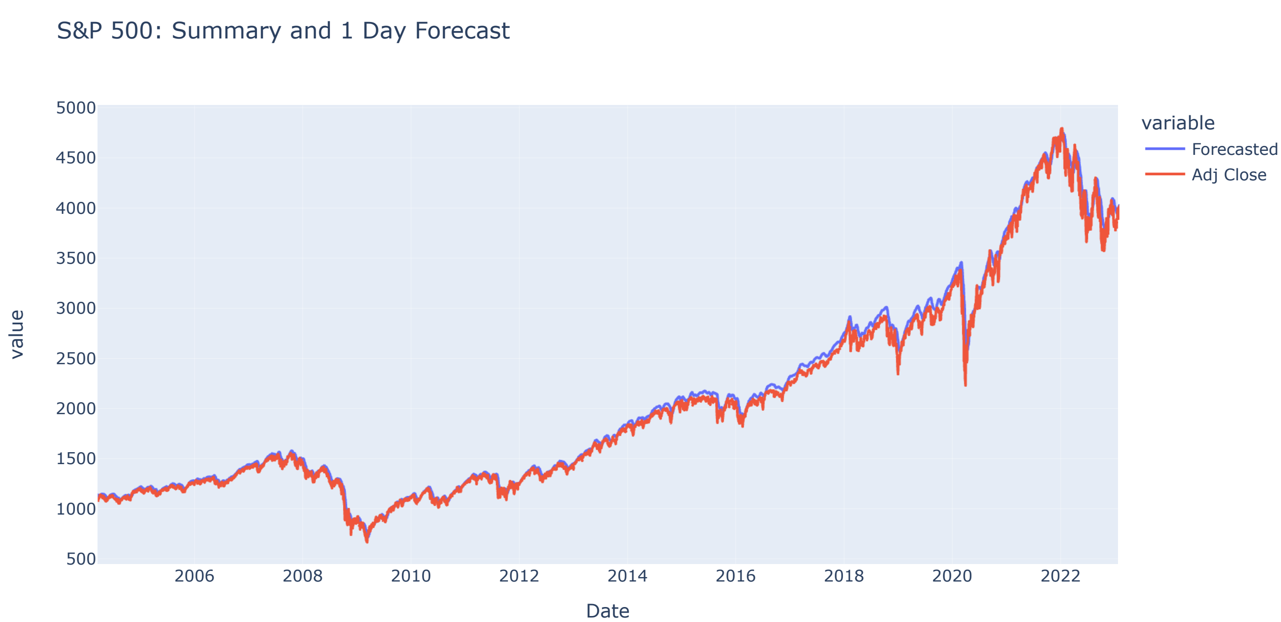
In [161]:
```
test_pred = model.predict(x_test)
test_pred = scaler.inverse_transform(test_pred)
```

```
149/149 [==============================] - 1s 5ms/step
```

In [162]:
```
summary_df = test_df[n_features:-1].reset_index()
summary_df['Forecasted'] = test_pred
```

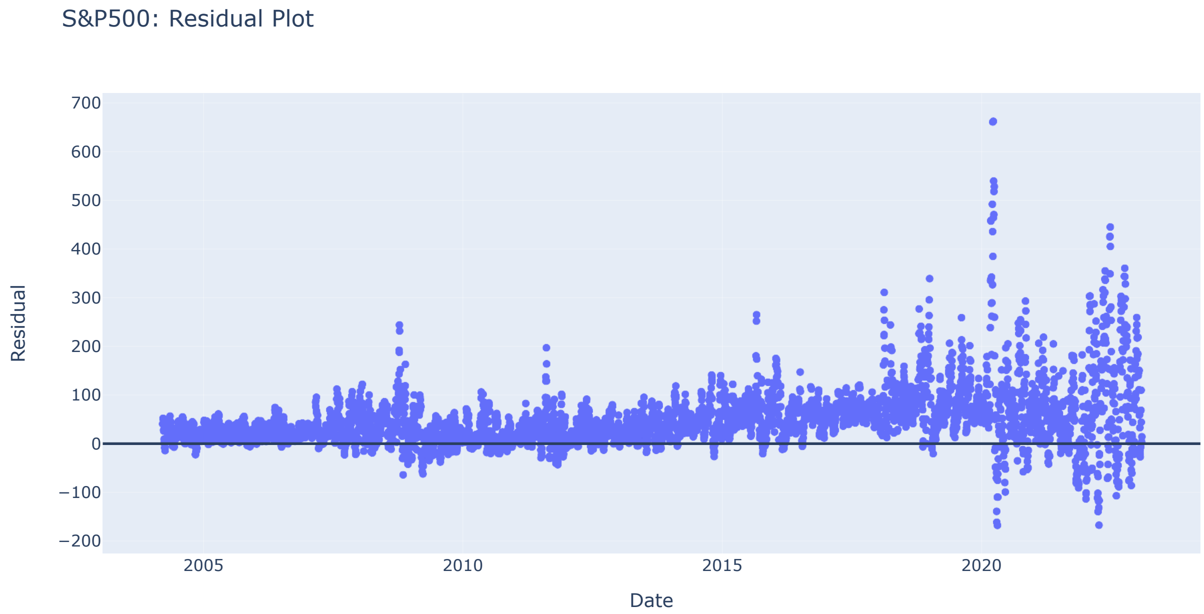
### Analysis

In [163]:
```
fig = px.line(summary_df, x="Date", y=["Forecasted", 'Adj Close'], title='S&P 500: Summary and 1 Day Forecast')
fig.show()
```




The plot tells us that the forecasted price (blue) and true price (red) is overlapped. There are small deviations on a day-to-day basis once you zoom in, but the **model was good at predicting market trends overall.**

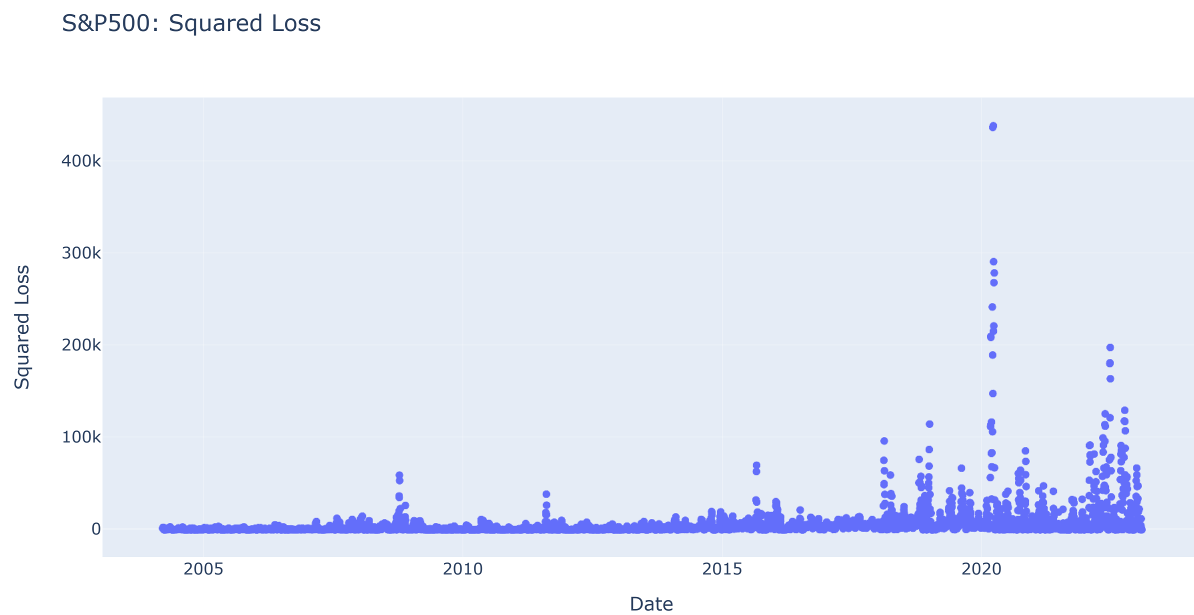
In [164]:
```
# Residual Plot
summary_df['Residual'] = summary_df['Forecasted'] - summary_df['Adj Close']
fig = px.scatter(summary_df,
          x='Date', y='Residual', title = 'S&P500: Residual Plot')
fig.add_hline(y=0)
fig.show()
```




In [165]:
```
# RMSE
summary_df['Squared Loss'] = (summary_df['Forecasted'] - summary_df['Adj Close'])**2
RMSE_pre = np.sqrt(summary_df['Squared Loss'][summary_df['Date']<'2018'].mean())
RMSE_post = np.sqrt(summary_df['Squared Loss'][summary_df['Date']>='2018'].mean())
RMSE_pre,RMSE_post
```

Out[165]:
```
(47.29226617199417, 126.4915178773102)
```

In [166]:
```
# Plot Square Loss
px.scatter(summary_df, x='Date', y='Squared Loss', title = 'S&P500: Squared Loss')
```



In [167]:
```
# Ten Losses
summary_df.sort_values('Squared Loss', ascending=False)[0:10]
```

Out[167]:

| | Date | Open | High | Low | Close | Adj Close | Volume | Forecasted | Residual | Squared Loss |
|---|---|---|---|---|---|---|---|---|---|---|
| **4024** | 2020-03-16 | 2508.590088 | 2562.979980 | 2380.939941 | 2386.129883 | 2386.129883 | 7805450000 | 3048.278564 | 662.148682 | 438440.876598 |
| **4022** | 2020-03-12 | 2630.860107 | 2660.949951 | 2478.860107 | 2480.639893 | 2480.639893 | 8850810000 | 3141.513672 | 660.873779 | 436754.152162 |
| **4026** | 2020-03-18 | 2436.500000 | 2453.570068 | 2280.520020 | 2398.100098 | 2398.100098 | 8799300000 | 2937.523438 | 539.423340 | 290977.539568 |
| **4029** | 2020-03-23 | 2290.709961 | 2300.729980 | 2191.860107 | 2237.399902 | 2237.399902 | 7411380000 | 2765.363525 | 527.963623 | 278745.587261 |
| **4028** | 2020-03-20 | 2431.939941 | 2453.010010 | 2295.560059 | 2304.919922 | 2304.919922 | 9053950000 | 2822.830078 | 517.910156 | 268230.929947 |
| **4019** | 2020-03-09 | 2863.889893 | 2863.889893 | 2734.429932 | 2746.560059 | 2746.560059 | 8441290000 | 3238.299316 | 491.739258 | 241807.497674 |
| **4027** | 2020-03-19 | 2393.479980 | 2466.969971 | 2319.780029 | 2409.389893 | 2409.389893 | 7956100000 | 2879.835938 | 470.446045 | 221319.481183 |
| **4025** | 2020-03-17 | 2425.659912 | 2553.929932 | 2367.040039 | 2529.189941 | 2529.189941 | 8370250000 | 2993.483154 | 464.293213 | 215568.187536 |
| **4012** | 2020-02-27 | 3062.540039 | 3097.070068 | 2977.389893 | 2978.760010 | 2978.760010 | 7064710000 | 3436.890625 | 458.130615 | 209883.660615 |
| **4013** | 2020-02-28 | 2916.899902 | 2959.719971 | 2855.840088 | 2954.219971 | 2954.219971 | 8569570000 | 3411.280029 | 457.060059 | 208903.897162 |

**Thoughts**

- The above scatterplot plots the **residual** and **squared loss** of our model respectably. Notice that the model has high loss everytime there is a economic turmoil (like in the late 90s and 2009s). However, the model unequivocally had the **highest loss during lockdown**. Although the model was able to predict trends even during the lockdown, it appears like the model **failed to predict the volatility of the market especially as the beginning of the pandemic.**
- From the table, we can see that the **ten highest model loss occured between February and March of 2020.**

**Other notes**

- When we used an 80:20 train-test split, the results were slightly better than 65:35. It might be because the training data learned from the dot-com bubble and was better equipped to predict the pandemic market.
- In general, the residual plots tended to overvalue. However, the spread appears random, which means the error results from noise and not systematic error.

## Part 2) ~~Tech Stocks:~~ **Debug*

I want to see how the model would perform for the tech sector specifically, seeing how growth in the tech sector was even more pronounced during the pandemic. NASDAQ 100 Technology Sector (NDXT) is an index composed of tech companies like Alphabet, Apple, and Meta (Facebook).

In [183]:
```
# df = yf.download('^NDXT')
```

In [184]:
```
# split_ratio = 0.75
# train_df, test_df = split(df, split_ratio)
# scaler = MinMaxScaler(feature_range = (0,1))
```

In [185]:
```
# n_features = 60

# # Normalize training and test data
# train_scaled = scaler.fit_transform(train_df['Adj Close'].values.reshape(-1,1))
# test_scaled = scaler.fit_transform(test_df['Adj Close'].values.reshape(-1,1))

# x_train, y_train = offset(train_scaled, n_features)
# x_test, y_test = offset(test_scaled, n_features)
```

In [186]:
```
# model = build_model()
# model.compile(loss='mean_squared_error', optimizer='adam')

# model.fit(x_train, y_train, epochs=6,
#           batch_size = 32)
```

In [187]:
```
# test_pred = model.predict(x_test)
# test_pred = scaler.inverse_transform(test_pred)
```

In [188]:
```
# summary_df = test_df[n_features:-1].reset_index()
# summary_df['Forecasted'] = test_pred
```

In [189]:
```
# fig = px.line(summary_df, x="Date", y=["Forecasted", 'Adj Close'], title='NDXT: Summary and Forecast')
# fig.show()
```

In [190]:
```
# # RMSE
# summary_df['Squared Loss'] = (summary_df['Forecasted'] - summary_df['Adj Close'])**2
# RMSE = np.sqrt(summary_df['Squared Loss'].mean())
# RMSE
```

In [182]:
```
# # Plot Error
# summary_df['Residual'] = summary_df['Forecasted'] - summary_df['Adj Close']
# fig = px.scatter(summary_df[summary_df['Date'] > '2005'],
#            x='Date', y='Residual', title = 'S&P500: Residual Scatterplot')
# fig.add_hline(y=0)
# fig.show()
```

# # Part 3) Apple (AAPL)

One of the most robust stocks in the tech industry, AAPL stocks tripled in value during the pandemic.

In [168]:
```
df = yf.download('AAPL')
```

```
[*********************100%***********************]  1 of 1 completed
```

In [169]:
```
split_ratio = 0.65
train_df, test_df = split(df, split_ratio)
scaler = MinMaxScaler(feature_range = (0,1))
```

In [170]:
```
n_features = 60

# Normalize training and test data
train_scaled = scaler.fit_transform(train_df['Adj Close'].values.reshape(-1,1))
test_scaled = scaler.fit_transform(test_df['Adj Close'].values.reshape(-1,1))

x_train, y_train = offset(train_scaled, n_features)
x_test, y_test = offset(test_scaled, n_features)
```

In [171]:
```
model = build_model()
model.compile(loss='mean_squared_error', optimizer='adam')

model.fit(x_train, y_train, epochs=6,
          batch_size = 32)
```

```
Epoch 1/6
214/214 [==============================] - 8s 23ms/step - loss: 0.0012
Epoch 2/6
214/214 [==============================] - 5s 23ms/step - loss: 6.2808e-04
Epoch 3/6
214/214 [==============================] - 5s 23ms/step - loss: 6.3165e-04
Epoch 4/6
214/214 [==============================] - 5s 23ms/step - loss: 4.4638e-04
Epoch 5/6
214/214 [==============================] - 5s 23ms/step - loss: 4.6732e-04
Epoch 6/6
214/214 [==============================] - 5s 23ms/step - loss: 5.0887e-04
```

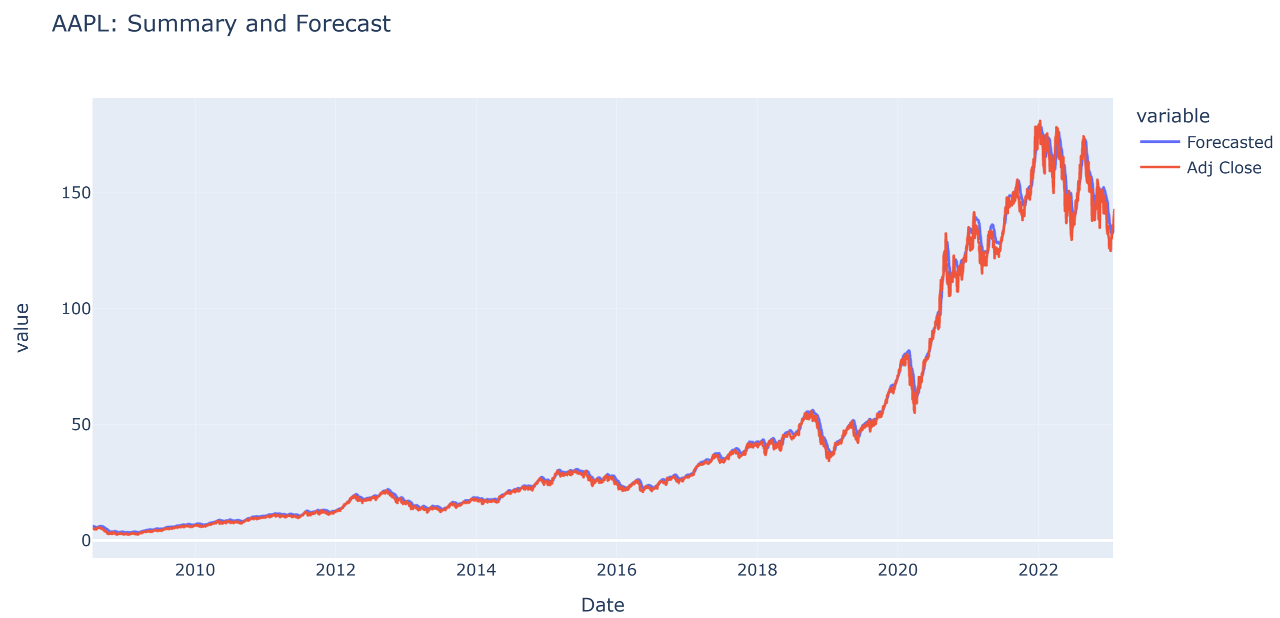
Out[171]:
```
<keras.callbacks.History at 0x1b61a9c1c70>
```

In [172]:
```
test_pred = model.predict(x_test)
test_pred = scaler.inverse_transform(test_pred)
```

```
115/115 [==============================] - 1s 8ms/step
```

In [173]:
```
summary_df = test_df[n_features:-1].reset_index()
summary_df['Forecasted'] = test_pred
```

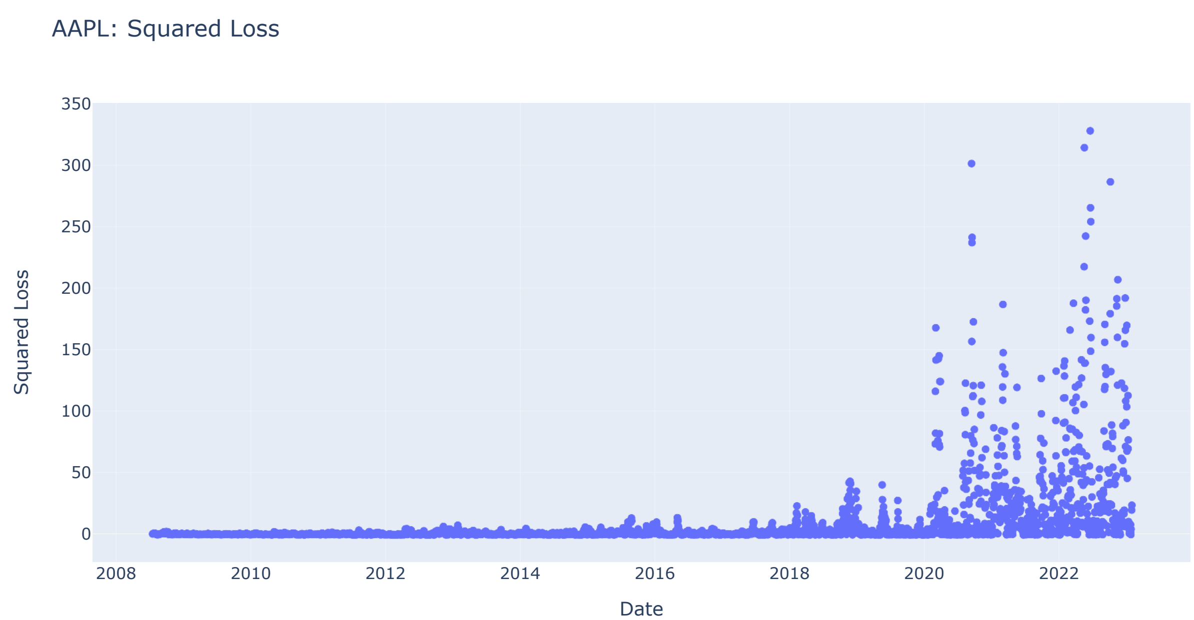
In [174]:
```
fig = px.line(summary_df, x="Date", y=["Forecasted", 'Adj Close'], title='AAPL: Summary and Forecast')
fig.show()
```



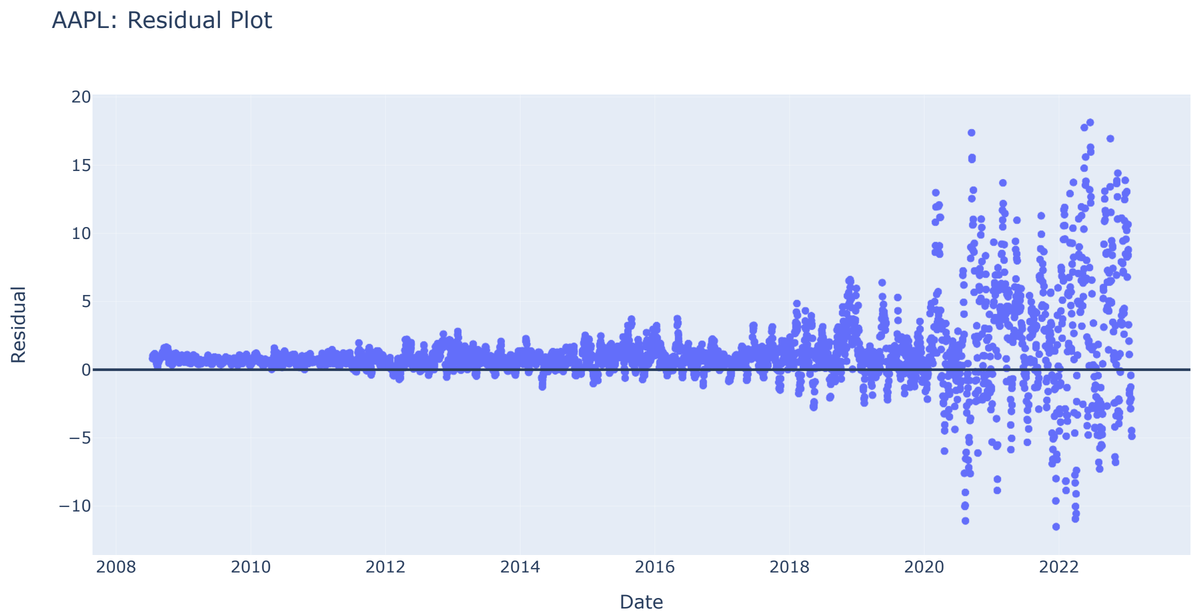

In [178]:
```
# RMSE
summary_df['Residual'] = summary_df['Forecasted'] - summary_df['Adj Close']
summary_df['Squared Loss'] = (summary_df['Forecasted'] - summary_df['Adj Close'])**2
RMSE = np.sqrt(summary_df['Squared Loss'].mean())
RMSE
```

Out[178]: 2.897211348505621

In [179]:
```
# Plot Square Error
px.scatter(summary_df, x='Date', y='Squared Loss', title = 'AAPL: Squared Loss')
```



In [180]:
```
# Residual Plot
fig = px.scatter(summary_df,
           x='Date', y='Residual', title = 'AAPL: Residual Plot')
fig.add_hline(y=0)
fig.show()
```




The plots are very similar to S&P500. While the model is good at predicting overall trends, it had difficulty forecasting patterns during the pandemic.

## Conclusion

During normal times, the model was a fantastic indicator of the market, with very little loss and predicting the market trends quite accurately. However, during the pandemic, we saw inflated loss values, especially at the onset of the lockdown. Although the model remains a good indicator on average, its predictions should be taken with a grain of salt during irregular times