Ethereum Dataset

```
import pandas as pd
from google.colab import drive
drive.mount('/content/drive')
df_ethereum_clean = pd.read_csv("/content/drive/MyDrive/ethereum.csv")
df_ethereum_clean.head()
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
```

	Date	% Day Change ETH	Max Difference ETH	Close ETH
0	2017-11-09	0.039654	22.395996	320.884003
1	2017-11-10	-0.066791	30.175995	299.252991
2	2017-11-11	0.053904	21.261017	314.681000
3	2017-11-12	-0.021551	20.640015	307.907990
4	2017-11-13	0.031564	21.390015	316.716003

Baseline Ethereum Model Process

Source Used

https://towardsdatascience.com/the-complete-guide-to-time-series-forecasting-using-sklearn-pandas-and-numpy-7694c90e45c1

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
```

Cleaning the data

```
df_ethereum_clean

df = df_ethereum_clean.dropna().copy()
df
```

	Date	% Day Change ETH	Max Difference ETH	Close ETH

1	2017-11-10	-0.066791	30.175995	299.252991
2	2017-11-11	0.053904	21.261017	314.681000
3	2017-11-12	-0.021551	20.640015	307.907990
4	2017-11-13	0.031564	21.390015	316.716003
...	...	...	...	...
1825	2022-11-08	-0.150156	315.356690	1332.835571
1826	2022-11-09	-0.174742	252.457885	1100.169800
1827	2022-11-10	0.181216	248.668579	1299.464600
1828	2022-11-11	-0.008978	90.965698	1287.221069
1830	2022-11-13	0.005559	7.459595	1262.385376

1830 rows × 4 columns

```
df["ETH SHIFT"] = df["Close ETH"].shift(-1)
```

Shift all the columns so the prediction would be the next column

Building our train and test datas our train would be 2019-2021, and our test data would be 2021-2020

```
train = df[700:1450]
test = df[1450:1900]
test = test.drop(test.tail(1).index)
```

```
train
```

	Date	% Day Change ETH	Max Difference ETH	Close ETH	ETH SHIFT
700	2019-10-10	-0.007927	5.590454	191.659668	182.569687
701	2019-10-11	-0.048130	13.656967	182.569687	180.826645
702	2019-10-12	-0.009354	6.320053	180.826645	182.075150
703	2019-10-13	0.006711	4.758758	182.075150	186.960907
704	2019-10-14	0.027097	5.641342	186.960907	181.406067
...	...	...	...	...	...
1445	2021-10-24	-0.020124	218.609131	4087.903076	4217.876953

1446	2021-10-25	0.032673	164.632324	4217.876953	4131.102051
1447	2021-10-26	-0.020448	182.307618	4131.102051	3930.257324
1448	2021-10-27	-0.048864	368.895508	3930.257324	4287.318848
1449	2021-10-28	0.092362	387.444824	4287.318848	4414.746582

750 rows × 5 columns

Make our test data based on the most reecent year 2022 and as you see predictions are shifted

```
test = test.copy()
test["baseline_predictions_ETH"] = test["Close ETH"]
test
```

	Date	% Day Change ETH	Max Difference ETH	Close ETH	ETH SHIFT	baseline_predictic
1450	2021-10-29	0.029394	184.027832	4414.746582	4325.650391	4414
1451	2021-10-30	-0.020070	174.354492	4325.650391	4288.074219	4325
1452	2021-10-31	-0.008020	215.433594	4288.074219	4324.626953	4288
1453	2021-11-01	0.008491	216.355469	4324.626953	4584.798828	4324
1454	2021-11-02	0.060682	311.261719	4584.798828	4607.193848	4584
...	...	...	...	...	...	
1824	2022-11-07	-0.002179	54.059815	1568.591309	1332.835571	1568
1825	2022-11-08	-0.150156	315.356690	1332.835571	1100.169800	1332
1826	2022-11-09	-0.174742	252.457885	1100.169800	1299.464600	1100
1827	2022-11-10	0.181216	248.668579	1299.464600	1287.221069	1299
1828	2022-11-11	-0.008978	90.965698	1287.221069	1262.385376	1287

Function that basically shifts more than one column

```
def window_input(window_length: int, data: pd.DataFrame) -> pd.DataFrame:

    df = data.copy()

    i = 1
    while i < window_length:
        df[f'x_{i}'] = df['Close ETH'].shift(-i)
```

```
        i = i + 1

    if i == window_length:
        df['y'] = df['Close ETH'].shift(-i)

    # Drop rows where there is a NaN
    df = df.dropna(axis=0)

    return df
```

Shifts the data 4 rows

```
new_df = window_input(5, df_ethereum_clean)
new_df
```

	Date	% Day Change ETH	Max Difference ETH	Close ETH	x_1	x_2	
0	2017-11-09	0.039654	22.395996	320.884003	299.252991	314.681000	307.90
1	2017-11-10	-0.066791	30.175995	299.252991	314.681000	307.907990	316.71
2	2017-11-11	0.053904	21.261017	314.681000	307.907990	316.716003	337.63
3	2017-11-12	-0.021551	20.640015	307.907990	316.716003	337.631012	333.35
4	2017-11-13	0.031564	21.390015	316.716003	337.631012	333.356995	330.92
...	...	...	...	...	...	...	
1819	2022-11-02	-0.037851	106.165894	1519.711792	1531.541748	1645.093384	1627.96
1820	2022-11-03	0.007776	39.657959	1531.541748	1645.093384	1627.968018	1572.23
1821	2022-11-04	0.074243	132.066284	1645.093384	1627.968018	1572.234741	1568.59
1822	2022-11-05	-0.010448	34.522217	1627.968018	1572.234741	1568.591309	1332.83
1823	2022-11-06	-0.034195	61.897705	1572.234741	1568.591309	1332.835571	1100.16

Splitting our data into testing and training data

```
from sklearn.model_selection import train_test_split

X = new_df[['Close ETH', 'x_1', 'x_2', 'x_3', 'x_4']].values
y = new_df['y'].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_sta
```

Applying a Baseline Model Predicts Model to predict the values

Applying a Baseline Model Predictor Model to predict the values

```
baseline_pred = []

for row in X_test:
    baseline_pred.append(np.mean(row))
```

Function to output a model of predictions

```
def window_input_output(input_length: int, output_length: int, data: pd.DataFrame)

    df = data.copy()

    i = 1
    while i < input_length:
        df[f'x_{i}'] = df["Close ETH"].shift(-i)
        i = i + 1

    j = 0
    while j < output_length:
        df[f'y_{j}'] = df["Close ETH"].shift(-output_length-j)
        j = j + 1

    df = df.dropna(axis=0)

    return df
```

Predicting a sequence of 365 values or a year

```
seq_df = window_input_output(365, 365, df_ethereum_clean)
seq_df
```

```
<ipython-input-112-5709ff65f261>:7: PerformanceWarning: DataFrame is highly f
  df[f'x_{i}'] = df["Close ETH"].shift(-i)
<ipython-input-112-5709ff65f261>:12: PerformanceWarning: DataFrame is highly
  df[f'y_{j}'] = df["Close ETH"].shift(-output_length-j)
```

	Date	% Day Change ETH	Max Difference ETH	Close ETH	x_1	x_2	x_3
0	2017-11-09	0.039654	22.395996	320.884003	299.252991	314.681000	307.907990
1	2017-11-10	-0.066791	30.175995	299.252991	314.681000	307.907990	316.716003
2	2017-11-11	0.053904	21.261017	314.681000	307.907990	316.716003	337.631012
3	2017-11-12	-0.021551	20.640015	307.907990	316.716003	337.631012	333.356995
4	2017-11-13	0.031564	21.390015	316.716003	337.631012	333.356995	330.924011

...	...	...	...	...	...	...	...
1095	2020-11-08	0.040935	24.626679	453.554779	444.163055	449.679626	462.960541
1096	2020-11-09	-0.020749	22.185730	444.163055	449.679626	462.960541	461.005280
1097	2020-11-10	0.012413	14.158234	449.679626	462.960541	461.005280	474.626434
1098	2020-11-11	0.029534	24.053924	462.960541	461.005280	474.626434	460.149841
1099	2020-11-12	-0.004221	15.605408	461.005280	474.626434	460.149841	447.559082

1100 rows × 733 columns

Splitting the data again

```
X_cols = [col for col in seq_df.columns if col.startswith('x')]

X_cols.insert(0, "Close ETH")

y_cols = [col for col in seq_df.columns if col.startswith('y')]
X_train = seq_df[X_cols][:-2].values
y_train = seq_df[y_cols][:-2].values

X_test = seq_df[X_cols][-2:].values
y_test = seq_df[y_cols][-2:].values
```

```
X_test[1][::-1]
```

```
array([4730.384277, 4636.174316, 4735.068848, 4812.087402, 4620.554688,
       4521.581055, 4486.243164, 4537.324219, 4607.193848, 4584.798828,
       4324.626953, 4288.074219, 4325.650391, 4414.746582, 4287.318848,
       3930.257324, 4131.102051, 4217.876953, 4087.903076, 4171.663574,
       3970.181885, 4054.322754, 4155.992188, 3877.650879, 3748.760254,
       3847.104492, 3830.38208 , 3862.634766, 3786.01416 , 3606.20166 ,
       3492.573242, 3545.354004, 3425.852783, 3575.716797, 3563.759277,
       3587.974854, 3580.562012, 3518.518555, 3380.089111, 3418.358643,
       3391.694336, 3307.516113, 3001.678955, 2853.143311, 2807.296631
```

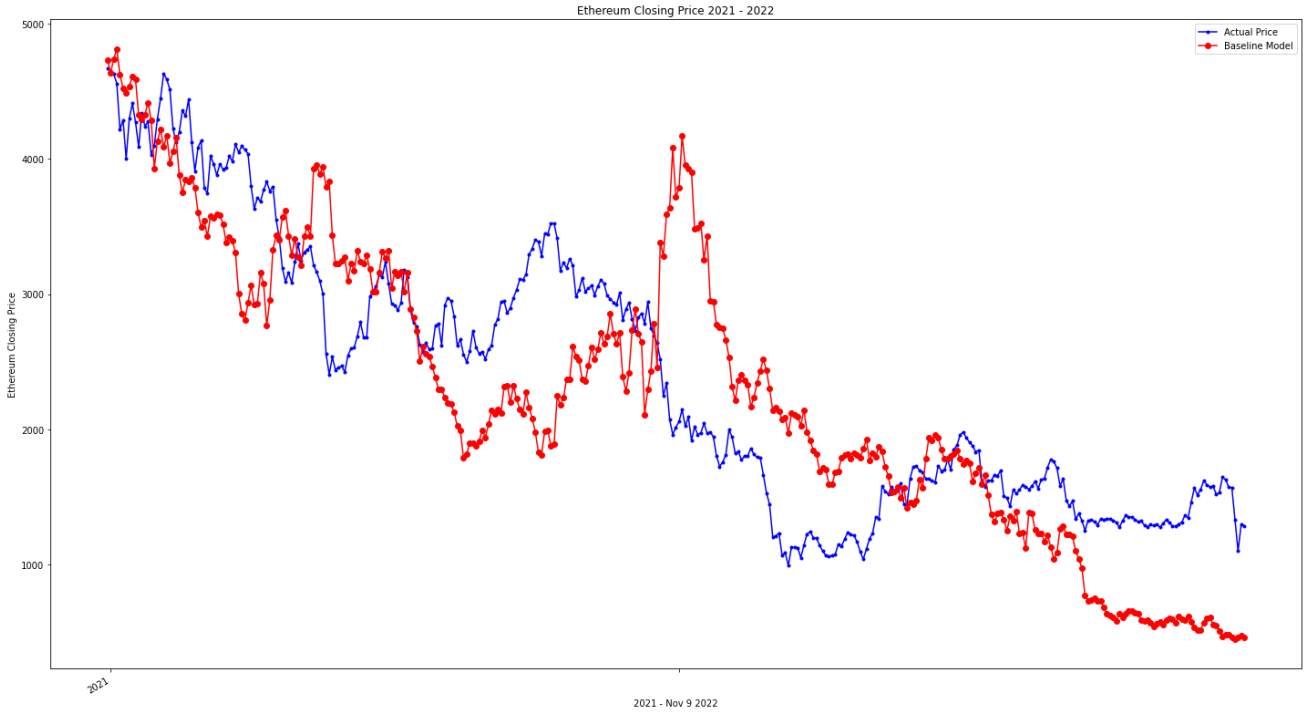
```
3391.094330, 3307.510113, 3001.078955, 2855.145311, 2807.290631,
2934.138916, 3062.265381, 2925.565674, 2931.669189, 3155.523682,
3077.86792 , 2764.431152, 2958.993408, 3329.447998, 3432.018311,
3398.538818, 3571.294922, 3615.282715, 3429.169678, 3285.511719,
3410.134521, 3270.278076, 3211.505859, 3427.340088, 3497.315186,
3426.394287, 3928.379395, 3952.133545, 3887.828369, 3940.614746,
3790.98999 , 3834.828125, 3433.732666, 3224.374268, 3227.002686,
3244.40332 , 3270.60083 , 3100.325439, 3224.915283, 3172.456299,
3319.257324, 3242.115479, 3226.083984, 3286.935303, 3182.702148,
3020.089844, 3014.845947, 3156.509521, 3310.50415 , 3265.443359,
3322.21167 , 3043.414307, 3164.245117, 3141.691162, 3167.856201,
3013.732666, 3157.23877 , 2890.94165 , 2827.328857, 2724.619873,
2502.349609, 2610.15332 , 2561.852051, 2536.209961, 2466.961426,
2380.956787, 2296.54541 , 2298.333496, 2233.366699, 2191.373779,
2189.21875 , 2124.776611, 2025.202759, 1990.970825, 1787.510742,
1817.296631, 1895.552124, 1898.825195, 1880.382935, 1911.175659,
1994.331299, 1940.083984, 2036.721069, 2139.664795, 2111.403564,
2146.692383, 2120.026367, 2315.161865, 2324.679443, 2198.58252 ,
2321.724121, 2226.114258, 2150.040283, 2113.605469, 2274.547607,
2160.768311, 2079.657471, 1978.894653, 1829.239258, 1813.217285,
1988.456299, 1989.736328, 1874.950073, 1888.44751 , 2246.364502,
2178.499023, 2231.733154, 2372.001953, 2367.663574, 2610.936768,
2537.891113, 2508.391602, 2372.484375, 2353.768799, 2471.518555,
2608.26709 , 2517.438721, 2590.263184, 2715.092773, 2630.576904,
2688.195068, 2855.126465, 2706.125   , 2633.518311, 2714.945313,
2390.30542 , 2279.51416 , 2419.90625 , 2736.488525, 2888.69873 ,
2706.628906, 2643.591064, 2109.579834, 2295.705566, 2430.621338,
2784.294189, 2460.679199, 3380.070068, 3282.397705, 3587.506104,
3638.12207 , 4079.057373, 3715.148438, 3785.848633, 4168.701172,
3952.293945, 3928.844727, 3902.647705, 3484.729004, 3490.880371,
3522.783203, 3253.629395, 3431.086182, 2952.056152, 2945.892822,
2773.207031, 2756.876953, 2746.380127, 2662.865234, 2534.481689,
2316.05957 , 2211.625732, 2363.586182, 2403.535156, 2364.751709,
2330.210938, 2166.188721, 2237.136963, 2344.89502 , 2431.946533,
2519.116211, 2435.10498 , 2299.187744, 2139.353271, 2157.656982,
2135.942139, 2072.108887, 2088.57373 , 1971.077271, 2118.378906,
2107.887207, 2093.122803, 2028.422485, 2143.225586, 1977.276855,
1918.362061, 1846.033691, 1819.684937, 1691.355957, 1716.494629,
1702.842041, 1595.359253, 1593.413452, 1678.650146, 1691.333984,
1788.217041, 1812.634644, 1817.624146, 1782.855103, 1823.449341,
1806.971802, 1791.702271, 1854.564331, 1924.685425, 1772.102417,
1826.194946, 1799.16626 , 1868.048828, 1834.727905, 1723.153809,
1654.741577, 1533.275024, 1541.914307, 1575.853149, 1492.608765,
1564.707642, 1416.04895 , 1459.973145, 1446.033691, 1475.703735,
1626.575684, 1570.203979, 1781.99292 , 1935.601074, 1919.534058,
1960.164795, 1937.449219, 1848.458252, 1781.067505, 1779.791016,
1805.084106, 1814.109863, 1843.532593, 1783.797974, 1744.243408,
1768.035034, 1746.616821, 1614.227783, 1677.846802, 1718.650879,
1594.762695, 1660.909546, 1515.193726, 1369.040527, 1314.986206,
1376.115479, 1382.522827, 1332.492188, 1253.187134, 1357.058105,
```

Plotted the data showing the Baseline Model vs Actual Price

```
fig, ax = plt.subplots(figsize=(20, 11))
```

```
ax.plot(np.arange(0, 365, 1), y_test[1], marker='.', color='blue', label='Actual Pr
ax.plot(np.arange(0, 365, 1), X_test[1][::-1], marker='o', color='red', label='Base

ax.set_title("Ethereum Closing Price 2021 - 2022")
ax.set_xlabel('2021 - Nov 9 2022')
ax.set_ylabel('Ethereum Closing Price')
plt.xticks(np.arange(1, 365, 182), np.arange(2021, 2022, 1))
plt.legend(loc=1)
fig.autofmt_xdate()
plt.tight_layout()
```



Colab paid products - Cancel contracts here