

## ▼ Background

Prior to beginning this project, I knew nothing about blockchain nor Ethereum specifically. I had heard market or dove deeper into how it works other than just knowing that it is out there. I am taking the opportunity to learn more about blockchain and Ethereum cryptocurrency. Looking at the data and, along the way, learn what makes the market tick. This project will be examining the Ethereum blockchain from the month of October in both 2018 and 2019. Comparing these attributes within each month and analyzing the data to find trends and patterns. This will provide analysis opportunities for me to understand exactly how each attribute relates to the other and how a change in one attribute may affect the other. I believe the information discovered during this project will not only be helpful for me to understand the Ethereum blockchain but also for anyone else who may be interested in learning more about the topics. Not only having an understanding of the Ethereum blockchain, but also understanding the relationships between these attributes proves valuable to investors and miners of Ethereum cryptocurrency. By examining the data, I will be able to look ahead in the market and predict future trends.

## ▼ Data

The data I will be examining in this project comes from the Google Bigquery dataset called crypto\_ethereum. For the purposes of understanding the blockchain and the Ethereum cryptocurrency, I will focus on the blocks, contracts, and transactions table. I wish I could use data from the other 4 tables, but they take a very long time to load. I will add on a week in October examination of the 4 tables for which querying the whole dataset is not feasible for my purposes.

Within the blocks table, I chose the attributes of the date, the block number, the miner activity, the difficulty experienced on the block up to the time of the transaction in question, the size of the block, the gas used by the transaction, and the transaction count.

Within the contracts table, I chose the attributes of the date and block number that the contract was created, whether or not the contract was an erc20 contract or an erc721 contract.

And lastly within the transaction table, I chose the attributes of the date and the block number the transaction was created in, the receiver address of the transaction, the value, the gas price at the time of the transaction, whether a transaction was successful or not, the gas used by the transaction, and the status of the transaction (either a 1 for a success or 0 for failure).

## ▼ Setting up Big Query Connection

```
from google.colab import auth  
auth.authenticate_user()
```

```
print('Authenticated')
```



```
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
import numpy as np
import pandas as pd
import datetime as dt
import plotly.express as px
import plotly.graph_objects as go
import os
import seaborn as sns
%matplotlib inline
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.tree import export_graphviz
import pydotplus

project_id = 'data-470-project'

from google.cloud import bigquery
client = bigquery.Client(project=project_id)
```

## ▼ Final Plan Outline

### ▼ *Blocks Table*

Query Google BigQuery

```
blocks_18 = client.query('''
SELECT DATE(timestamp) as date, number, miner, difficulty, total_difficulty, size, gas_li
FROM `bigquery-public-data.crypto_ethereum.blocks`
WHERE DATE(timestamp) >= "2018-10-01" AND DATE(timestamp) < "2018-10-31" ''').to_dataframe()

blocks_19 = client.query('''
SELECT DATE(timestamp) as date, number, miner, difficulty, total_difficulty, size, gas_li
FROM `bigquery-public-data.crypto_ethereum.blocks`
WHERE DATE(timestamp) >= "2019-10-01" AND DATE(timestamp) < "2019-10-31" ''').to_dataframe()
```

Concat serparte dfs into one df called blocks and add in dayofyear, year, and day of week columns

```

frames = [blocks_18, blocks_19]
blocks = pd.concat(frames)
blocks_num = blocks.groupby(by='date', as_index=False, sort = 'True').agg({'number': pd.Series})
blocks_min = blocks.groupby(by='date', as_index=False, sort = 'True').agg({'miner': pd.Series})
blocks_min = blocks_min[['miner']]
blocks_num['date'] = pd.to_datetime(blocks_num['date'], errors='coerce')
blocks_num['dayofyear'] = blocks_num['date'].dt.dayofyear
blocks_num['year'] = pd.DatetimeIndex(blocks_num['date']).year
blocks_num['dow'] = blocks_num['date'].dt.day_name()

blocks_diff = blocks.groupby(by='date', as_index=False, sort = 'True').agg({'difficulty': pd.Series})
blocks_diff = blocks_diff[['difficulty']]

blocks_gas = blocks.groupby(by='date', as_index=False, sort = 'True').agg({'gas_used': pd.Series})
blocks_gas = blocks_gas[['gas_used']]

blocks_trans = blocks.groupby(by='date', as_index=False, sort = 'True').agg({'transaction_count': pd.Series})
blocks_trans = blocks_trans[['transaction_count']]

```

Concat all calculations from above

```

blocks_comp = pd.concat([blocks_num, blocks_min, blocks_diff, blocks_gas, blocks_trans], axis=1)
blocks_comp.head()

```



## ▼ Plots

```

blocks_comp['year'] = blocks_comp['year'].astype(str)
fig = px.scatter_matrix(blocks_comp, dimensions=["number", "miner", "difficulty", "gas_used"])
fig.show()

fig = px.line(blocks_comp, x="dayofyear", y="number", color = "year", hover_name= "dow", title="Number of blocks per day of year by year")
fig.update_layout(
    xaxis=dict(
        tickvals=[1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260, 270, 280, 290, 300, 310, 320, 330, 340, 350, 360, 370, 380, 390, 400, 410, 420, 430, 440, 450, 460, 470, 480, 490, 500, 510, 520, 530, 540, 550, 560, 570, 580, 590, 600, 610, 620, 630, 640, 650, 660, 670, 680, 690, 700, 710, 720, 730, 740, 750, 760, 770, 780, 790, 800, 810, 820, 830, 840, 850, 860, 870, 880, 890, 900, 910, 920, 930, 940, 950, 960, 970, 980, 990, 1000, 1010, 1020, 1030, 1040, 1050, 1060, 1070, 1080, 1090, 1100, 1110, 1120, 1130, 1140, 1150, 1160, 1170, 1180, 1190, 1200, 1210, 1220, 1230, 1240, 1250, 1260, 1270, 1280, 1290, 1300, 1310, 1320, 1330, 1340, 1350, 1360, 1370, 1380, 1390, 1400, 1410, 1420, 1430, 1440, 1450, 1460, 1470, 1480, 1490, 1500, 1510, 1520, 1530, 1540, 1550, 1560, 1570, 1580, 1590, 1600, 1610, 1620, 1630, 1640, 1650, 1660, 1670, 1680, 1690, 1700, 1710, 1720, 1730, 1740, 1750, 1760, 1770, 1780, 1790, 1800, 1810, 1820, 1830, 1840, 1850, 1860, 1870, 1880, 1890, 1900, 1910, 1920, 1930, 1940, 1950, 1960, 1970, 1980, 1990, 2000, 2010, 2020, 2030, 2040, 2050, 2060, 2070, 2080, 2090, 2100, 2110, 2120, 2130, 2140, 2150, 2160, 2170, 2180, 2190, 2200, 2210, 2220, 2230, 2240, 2250, 2260, 2270, 2280, 2290, 2300, 2310, 2320, 2330, 2340, 2350, 2360, 2370, 2380, 2390, 2400, 2410, 2420, 2430, 2440, 2450, 2460, 2470, 2480, 2490, 2500, 2510, 2520, 2530, 2540, 2550, 2560, 2570, 2580, 2590, 2600, 2610, 2620, 2630, 2640, 2650, 2660, 2670, 2680, 2690, 2700, 2710, 2720, 2730, 2740, 2750, 2760, 2770, 2780, 2790, 2800, 2810, 2820, 2830, 2840, 2850, 2860, 2870, 2880, 2890, 2900, 2910, 2920, 2930, 2940, 2950, 2960, 2970, 2980, 2990, 3000, 3010, 3020, 3030, 3040, 3050, 3060, 3070, 3080, 3090, 3100, 3110, 3120, 3130, 3140, 3150, 3160, 3170, 3180, 3190, 3200, 3210, 3220, 3230, 3240, 3250, 3260, 3270, 3280, 3290, 3300, 3310, 3320, 3330, 3340, 3350, 3360, 3370, 3380, 3390, 3400, 3410, 3420, 3430, 3440, 3450, 3460, 3470, 3480, 3490, 3500, 3510, 3520, 3530, 3540, 3550, 3560, 3570, 3580, 3590, 3600, 3610, 3620, 3630, 3640, 3650, 3660, 3670, 3680, 3690, 3700, 3710, 3720, 3730, 3740, 3750, 3760, 3770, 3780, 3790, 3800, 3810, 3820, 3830, 3840, 3850, 3860, 3870, 3880, 3890, 3900, 3910, 3920, 3930, 3940, 3950, 3960, 3970, 3980, 3990, 4000, 4010, 4020, 4030, 4040, 4050, 4060, 4070, 4080, 4090, 4100, 4110, 4120, 4130, 4140, 4150, 4160, 4170, 4180, 4190, 4200, 4210, 4220, 4230, 4240, 4250, 4260, 4270, 4280, 4290, 4300, 4310, 4320, 4330, 4340, 4350, 4360, 4370, 4380, 4390, 4400, 4410, 4420, 4430, 4440, 4450, 4460, 4470, 4480, 4490, 4500, 4510, 4520, 4530, 4540, 4550, 4560, 4570, 4580, 4590, 4600, 4610, 4620, 4630, 4640, 4650, 4660, 4670, 4680, 4690, 4700, 4710, 4720, 4730, 4740, 4750, 4760, 4770, 4780, 4790, 4800, 4810, 4820, 4830, 4840, 4850, 4860, 4870, 4880, 4890, 4900, 4910, 4920, 4930, 4940, 4950, 4960, 4970, 4980, 4990, 5000, 5010, 5020, 5030, 5040, 5050, 5060, 5070, 5080, 5090, 5100, 5110, 5120, 5130, 5140, 5150, 5160, 5170, 5180, 5190, 5200, 5210, 5220, 5230, 5240, 5250, 5260, 5270, 5280, 5290, 5300, 5310, 5320, 5330, 5340, 5350, 5360, 5370, 5380, 5390, 5400, 5410, 5420, 5430, 5440, 5450, 5460, 5470, 5480, 5490, 5500, 5510, 5520, 5530, 5540, 5550, 5560, 5570, 5580, 5590, 5600, 5610, 5620, 5630, 5640, 5650, 5660, 5670, 5680, 5690, 5700, 5710, 5720, 5730, 5740, 5750, 5760, 5770, 5780, 5790, 5800, 5810, 5820, 5830, 5840, 5850, 5860, 5870, 5880, 5890, 5890, 5900, 5910, 5920, 5930, 5940, 5950, 5960, 5970, 5980, 5990, 6000, 6010, 6020, 6030, 6040, 6050, 6060, 6070, 6080, 6090, 6100, 6110, 6120, 6130, 6140, 6150, 6160, 6170, 6180, 6190, 6200, 6210, 6220, 6230, 6240, 6250, 6260, 6270, 6280, 6290, 6300, 6310, 6320, 6330, 6340, 6350, 6360, 6370, 6380, 6390, 6400, 6410, 6420, 6430, 6440, 6450, 6460, 6470, 6480, 6490, 6500, 6510, 6520, 6530, 6540, 6550, 6560, 6570, 6580, 6590, 6600, 6610, 6620, 6630, 6640, 6650, 6660, 6670, 6680, 6690, 6700, 6710, 6720, 6730, 6740, 6750, 6760, 6770, 6780, 6790, 6800, 6810, 6820, 6830, 6840, 6850, 6860, 6870, 6880, 6890, 6900, 6910, 6920, 6930, 6940, 6950, 6960, 6970, 6980, 6990, 7000, 7010, 7020, 7030, 7040, 7050, 7060, 7070, 7080, 7090, 7100, 7110, 7120, 7130, 7140, 7150, 7160, 7170, 7180, 7190, 7200, 7210, 7220, 7230, 7240, 7250, 7260, 7270, 7280, 7290, 7300, 7310, 7320, 7330, 7340, 7350, 7360, 7370, 7380, 7390, 7400, 7410, 7420, 7430, 7440, 7450, 7460, 7470, 7480, 7490, 7500, 7510, 7520, 7530, 7540, 7550, 7560, 7570, 7580, 7590, 7600, 7610, 7620, 7630, 7640, 7650, 7660, 7670, 7680, 7690, 7700, 7710, 7720, 7730, 7740, 7750, 7760, 7770, 7780, 7790, 7800, 7810, 7820, 7830, 7840, 7850, 7860, 7870, 7880, 7890, 7890, 7900, 7910, 7920, 7930, 7940, 7950, 7960, 7970, 7980, 7990, 8000, 8010, 8020, 8030, 8040, 8050, 8060, 8070, 8080, 8090, 8090, 8100, 8110, 8120, 8130, 8140, 8150, 8160, 8170, 8180, 8190, 8200, 8210, 8220, 8230, 8240, 8250, 8260, 8270, 8280, 8290, 8300, 8310, 8320, 8330, 8340, 8350, 8360, 8370, 8380, 8390, 8400, 8410, 8420, 8430, 8440, 8450, 8460, 8470, 8480, 8490, 8500, 8510, 8520, 8530, 8540, 8550, 8560, 8570, 8580, 8590, 8600, 8610, 8620, 8630, 8640, 8650, 8660, 8670, 8680, 8690, 8690, 8700, 8710, 8720, 8730, 8740, 8750, 8760, 8770, 8780, 8790, 8790, 8800, 8810, 8820, 8830, 8840, 8850, 8860, 8870, 8880, 8890, 8890, 8900, 8910, 8920, 8930, 8940, 8950, 8960, 8970, 8980, 8990, 9000, 9010, 9020, 9030, 9040, 9050, 9060, 9070, 9080, 9090, 9100, 9110, 9120, 9130, 9140, 9150, 9160, 9170, 9180, 9190, 9200, 9210, 9220, 9230, 9240, 9250, 9260, 9270, 9280, 9290, 9300, 9310, 9320, 9330, 9340, 9350, 9360, 9370, 9380, 9390, 9400, 9410, 9420, 9430, 9440, 9450, 9460, 9470, 9480, 9490, 9500, 9510, 9520, 9530, 9540, 9550, 9560, 9570, 9580, 9590, 9600, 9610, 9620, 9630, 9640, 9650, 9660, 9670, 9680, 9690, 9690, 9700, 9710, 9720, 9730, 9740, 9750, 9760, 9770, 9780, 9790, 9790, 9800, 9810, 9820, 9830, 9840, 9850, 9860, 9870, 9880, 9890, 9890, 9900, 9910, 9920, 9930, 9940, 9950, 9960, 9970, 9980, 9990, 10000, 10010, 10020, 10030, 10040, 10050, 10060, 10070, 10080, 10090, 10090, 10100, 10110, 10120, 10130, 10140, 10150, 10160, 10170, 10180, 10190, 10190, 10200, 10210, 10220, 10230, 10240, 10250, 10260, 10270, 10280, 10290, 10290, 10300, 10310, 10320, 10330, 10340, 10350, 10360, 10370, 10380, 10390, 10390, 10400, 10410, 10420, 10430, 10440, 10450, 10460, 10470, 10480, 10490, 10490, 10500, 10510, 10520, 10530, 10540, 10550, 10560, 10570, 10580, 10590, 10590, 10600, 10610, 10620, 10630, 10640, 10650, 10660, 10670, 10680, 10690, 10690, 10700, 10710, 10720, 10730, 10740, 10750, 10760, 10770, 10780, 10790, 10790, 10800, 10810, 10820, 10830, 10840, 10850, 10860, 10870, 10880, 10890, 10890, 10900, 10910, 10920, 10930, 10940, 10950, 10960, 10970, 10980, 10980, 10990, 10990, 11000, 11010, 11020, 11030, 11040, 11050, 11060, 11070, 11080, 11090, 11090, 11100, 11110, 11120, 11130, 11140, 11150, 11160, 11170, 11180, 11190, 11190, 11200, 11210, 11220, 11230, 11240, 11250, 11260, 11270, 11280, 11290, 11290, 11300, 11310, 11320, 11330, 11340, 11350, 11360, 11370, 11380, 11390, 11390, 11400, 11410, 11420, 11430, 11440, 11450, 11460, 11470, 11480, 11490, 11490, 11500, 11510, 11520, 11530, 11540, 11550, 11560, 11570, 11580, 11590, 11590, 11600, 11610, 11620, 11630, 11640, 11650, 11660, 11670, 11680, 11690, 11690, 11700, 11710, 11720, 11730, 11740, 11750, 11760, 11770, 11780, 11780, 11790, 11790, 11800, 11810, 11820, 11830, 11840, 11850, 11860, 11870, 11880, 11880, 11890, 11890, 11900, 11910, 11920, 11930, 11940, 11950, 11960, 11970, 11980, 11980, 11990, 11990, 12000, 12010, 12020, 12030, 12040, 12050, 12060, 12070, 12080, 12090, 12090, 12100, 12110, 12120, 12130, 12140, 12150, 12160, 12170, 12180, 12190, 12190, 12200, 12210, 12220, 12230, 12240, 12250, 12260, 12270, 12280, 12290, 12290, 12300, 12310, 12320, 12330, 12340, 12350, 12360, 12370, 12380, 12390, 12390, 12400, 12410, 12420, 12430, 12440, 12450, 12460, 12470, 12480, 12490, 12490, 12500, 12510, 12520, 12530, 12540, 12550, 12560, 12570, 12580, 12590, 12590, 12600, 12610, 12620, 12630, 12640, 12650, 12660, 12670, 12680, 12690, 12690, 12700, 12710, 12720, 12730, 12740, 12750, 12760, 12770, 12780, 12780, 12790, 12790, 12800, 12810, 12820, 12830, 12840, 12850, 12860, 12870, 12880, 12880, 12890, 12890, 12900, 12910, 12920, 12930, 12940, 12950, 12960, 12970, 12980, 12980, 12990, 12990, 13000, 13010, 13020, 13030, 13040, 13050, 13060, 13070, 13080, 13090, 13090, 13100, 13110, 13120, 13130, 13140, 13150, 13160, 13170, 13180, 13190, 13190, 13200, 13210, 13220, 13230, 13240, 13250, 13260, 13270, 13280, 13290, 13290, 13300, 13310, 13320, 13330, 13340, 13350, 13360, 13370, 13380, 13390, 13390, 13400, 13410, 13420, 13430, 13440, 13450, 13460, 13470, 13480, 13490, 13490, 13500, 13510, 13520, 13530, 13540, 13550, 13560, 13570, 13580, 13590, 13590, 13600, 13610, 13620, 13630, 13640, 13650, 13660, 13670, 13680, 13690, 13690, 13700, 13710, 13720, 13730, 13740, 13750, 13760, 13770, 13780, 13780, 13790, 13790, 13800, 13810, 13820, 13830, 13840, 13850, 13860, 13870, 13880, 13880, 13890, 13890, 13900, 13910, 13920, 13930, 13940, 13950, 13960, 13970, 13980, 13980, 13990, 13990, 14000, 14010, 14020, 14030, 14040, 14050, 14060, 14070, 14080, 14090, 14090, 14100, 14110, 14120, 14130, 14140, 14150, 14160, 14170, 14180, 14190, 14190, 14200, 14210, 14220, 14230, 14240, 14250, 14260, 14270, 14280, 14290, 14290, 14300, 14310, 14320, 14330, 14340, 14350, 14360, 14370, 14380, 14390, 14390, 14400, 14410, 14420, 14430, 14440, 14450, 14460, 14470, 14480, 14490, 14490, 14500, 14510, 14520, 14530, 14540, 14550, 14560, 14570, 14580, 14590, 14590, 14600, 14610, 14620, 14630, 14640, 14650, 14660, 14670, 14680, 14690, 14690, 14700, 14710, 14720, 14730, 14740, 14750, 14760, 14770, 14780, 14780, 14790, 14790, 14800, 14810, 14820, 14830, 14840, 14850, 14860, 14870, 14880, 14880, 14890, 14890, 14900, 14910, 14920, 14930, 14940, 14950, 14960, 14970, 14980, 14980, 14990, 14990, 15000, 15010, 15020, 15030, 15040, 15050, 15060, 15070, 15080, 15090, 15090, 15100, 15110, 15120, 15130, 15140, 15150, 15160, 15170, 15180, 15190, 15190, 15200, 15210, 15220, 15230, 15240, 15250, 15260, 15270, 15280, 15290, 15290, 15300, 15310, 15320, 15330, 15340, 15350, 15360, 15370, 15380, 15390, 15390, 15400, 15410, 15420, 15430, 15440, 15450, 15460, 15470, 15480, 15490, 15490, 15500, 15510, 15520, 15530, 15540, 15550, 15560, 15570, 15580, 15590, 15590, 15600, 15610, 15620, 15630, 15640, 15650, 15660, 15670, 15680, 15690, 15690, 15700, 15710, 15720, 15730, 15740, 15750, 15760, 15770, 15780, 15780, 15790, 15790, 15800, 15810, 15820, 15830, 15840, 15850, 15860, 15870, 15880, 15880, 15890, 15890, 15900, 15910, 15920, 15930, 15940, 15950, 15960, 15970, 15980, 15980, 15990, 15990, 16000, 16010, 16020, 16030, 16040, 16050, 16060, 16070, 16
```

```
title_text = "Day Of Year",
showline=True,
showgrid=False,
showticklabels=True,
linecolor='rgb(204, 204, 204)',
linewidth=2,
ticks='outside',
tickfont=dict(
    family='Arial',
    size=12,
    color='rgb(82, 82, 82)'),
),
yaxis=dict(
    title_text = "Block Count",
    showgrid=False,
    zeroline=False,
    showline=False,
    showticklabels=True,
),
margin=dict(
    autoexpand=False,
    l=100,
    r=100,
    t=110,
),
barmode = 'group',
legend_orientation = "h",
showlegend=True,
plot_bgcolor='white'
)
fig.show()
```



```
fig = px.line(blocks_comp, x="dayofyear", y="miner", color = "year", hover_name= "dow", tit
fig.update_layout(
    xaxis=dict(
        title_text = "Day Of Year",
        showline=True,
        showgrid=False,
        showticklabels=True,
        linecolor='rgb(204, 204, 204)',
        linewidth=2,
        ticks='outside',
        tickfont=dict(
            family='Arial',
            size=12,
            color='rgb(82, 82, 82)'
        ),
    ),
    yaxis=dict(
        title_text = "Miner Count",
        showgrid=False,
        zeroline=False,
        showline=False,
        showticklabels=True,
    ),
    margin=dict(
        autoexpand=False,
        l=100,
        r=100,
        t=110,
    ),
    barmode = 'group',
    legend_orientation = "h",
    showlegend=True,
    plot_bgcolor='white'
)
fig.show()
```



```
fig = px.line(blocks_comp, x="dayofyear", y="difficulty", color = "year", hover_name= "dow"
fig.update_layout(
    xaxis=dict(
        title_text = "Day Of Year",
        showline=True,
        showgrid=False,
        showticklabels=True,
        linecolor='rgb(204, 204, 204)',
        linewidth=2,
        ticks='outside',
        tickfont=dict(
            family='Arial',
            size=12,
            color='rgb(82, 82, 82)'
        ),
    ),
    yaxis=dict(
        title_text = "Average Difficulty",
        showgrid=False,
        zeroline=False,
```

```
    showline=False,
    showticklabels=True,
),
margin=dict(
    autoexpand=False,
    l=100,
    r=100,
    t=110,
),
barmode = 'group',
legend_orientation = "h",
showlegend=True,
plot_bgcolor='white'
)
fig.show()
```



```
fig = px.line(blocks_comp, x="dayofyear", y="gas_used", color = "year", hover_name= "dow",
fig.update_layout(
    title=
```

```
        title_text = "Day Of Year",
        showline=True,
        showgrid=False,
        showticklabels=True,
        linecolor='rgb(204, 204, 204)',
        linewidth=2,
        ticks='outside',
        tickfont=dict(
            family='Arial',
            size=12,
            color='rgb(82, 82, 82)'
        ),
    ),
    yaxis=dict(
        title_text = "Total Gas Used",
        showgrid=False,
        zeroline=False,
        showline=False,
        showticklabels=True,
    ),
    margin=dict(
        autoexpand=False,
        l=100,
        r=100,
        t=110,
    ),
    barmode = 'group',
    legend_orientation = "h",
    showlegend=True,
    plot_bgcolor='white'
)
fig.show()
```



```
fig = px.line(blocks_comp, x="dayofyear", y="transaction_count", color = "year", hover_name
fig.update_layout(
    xaxis=dict(
        title_text = "Day Of Year",
        showline=True,
        showgrid=False,
        showticklabels=True,
        linecolor='rgb(204, 204, 204)',
        linewidth=2,
        ticks='outside',
        tickfont=dict(
            family='Arial',
            size=12,
            color='rgb(82, 82, 82)'),
    ),
    yaxis=dict(
        title_text = "Transaction Count",
        showgrid=False,
        zeroline=False,
        showline=False,
        showticklabels=True,
    ),
    margin=dict(
        autoexpand=False,
        l=100,
        r=100,
        t=110,
    ),
    barmode = 'group',
    legend_orientation = "h",
    showlegend=True,
    plot_bgcolor='white'
)
fig.show()
```



## ▼ Random Forrest: Can we predict the day of the week?

```
labels = np.array(blocks_comp['dow'])  
labels[:5]
```



```
feature = blocks_comp.drop('date', axis = 1)  
feature_1 = feature.drop('dayofyear', axis = 1)  
features = feature_1.drop('year', axis = 1)  
features = features.drop('dow', axis = 1)
```

```
features.head()
```



```
rf = RandomForestClassifier(n_estimators = 1000, random_state = 42)
```

```
rf.fit(features, labels)
```

↳

```
predictions = rf.predict(features)
```

```
blocks_comp['predicted'] = predictions
```

```
blocks_comp.head()
```

↳

```
blocks_comp.count()
```

↳

```
blocks_comp[['dow']].count()
```

↳

```
blocks_comp.groupby('dow').count()['predicted']
```

↳

```
train_features, test_features, train_labels, test_labels = \
train_test_split(features, labels, test_size = 0.25, random_state = 42)
```

```
print("Size of the training data", train_features.shape)
print("Size of the test data", test_features.shape)
print("Size of the training labels", train_labels.shape)
print("Size of the test labels", test_labels.shape)
```

↳

```
rf = RandomForestClassifier(n_estimators = 1000, random_state = 42)
```

```
rf.fit(train_features, train_labels);
```

```
predictions = rf.predict(test_features)
```

```
print(predictions)
print(test_labels)
```

↳

```
conf_mat = pd.crosstab(predictions, test_labels, rownames=['Actual DOW'], \
                        colnames=['Predicted DOW'])
conf_mat
```

↳

```
rf.feature_importances_
```

↳

```
list(zip(train_features, rf.feature_importances_))
```

↳

```
feature_list = list(features.columns.values)
feature_list
```

↳

```
unique_pos = blocks_comp['dow'].unique()
unique_pos
target_names = list(unique_pos)
target_names
```

↳

```
estimator = rf.estimators_[0]
from sklearn.tree import export_graphviz
# Export as dot file
tree = export_graphviz(estimator, out_file='tree.dot',
```

```
feature_names = feature_list,
class_names = target_names,
rounded = True, proportion = False,
precision = 2, filled = True)

##  
# Convert to png using system command (requires Graphviz)
from subprocess import call
call(['dot', '-Tpng', 'tree.dot', '-o', 'tree.png', '-Gdpi=600'])

# Display in jupyter notebook
from IPython.display import Image
Image(filename = 'tree.png')
```

□→

I'm not too sure why this is coming out so large

## ▼ Contracts

Query Google BigQuery contracts table

```
contracts_18 = client.query('''
SELECT DATE(block_timestamp) as date, block_number, address, is_erc20, is_erc721
FROM `bigquery-public-data.crypto Ethereum.contracts`
WHERE DATE(block_timestamp) >= "2018-10-01" AND DATE(block_timestamp) < "2018-10-31"''').to

contracts_19 = client.query('''
SELECT DATE(block_timestamp) as date, block_number, address, is_erc20, is_erc721
FROM `bigquery-public-data.crypto Ethereum.contracts`
WHERE DATE(block_timestamp) >= "2019-10-01" AND DATE(block_timestamp) < "2019-10-31"''').to

frames = [contracts_18, contracts_19]
contracts = pd.concat(frames)
contracts_num = contracts.groupby(by='date', as_index=False, sort = 'True').agg({'block_num':sum})
contracts_20 = contracts.groupby(by='date', as_index=False, sort = 'True').agg({'is_erc20':sum})
contracts_721 = contracts.groupby(by='date', as_index=False, sort = 'True').agg({'is_erc721':sum})
contracts_num['date'] = pd.to_datetime(contracts_num['date'], errors='coerce')
contracts_num['dayofyear'] = contracts_num['date'].dt.dayofyear
contracts_num['year'] = pd.DatetimeIndex(contracts_num['date']).year
contracts_num['dow'] = contracts_num['date'].dt.day_name()
contracts_20 = contracts_20[['is_erc20']]
contracts_721 = contracts_721[['is_erc721']]
contracts_comp = pd.concat([contracts_num, contracts_20, contracts_721], axis=1, sort=False)

contracts_comp.head()
```



## ▼ Plots

```
fig = px.line(contracts_comp, x="dayofyear", y="block_number", color = "year", hover_name=
fig.update_layout(
    xaxis=dict(
        title_text = "Day Of Year",
        showline=True,
        showgrid=False,
        showticklabels=True,
        linecolor='rgb(204, 204, 204)',
        linewidth=2,
        ticks='outside',
        tickfont=dict(
            family='Arial',
            size=12,
            color='rgb(82, 82, 82)'),
    ),
    yaxis=dict(
        title_text = "Block Count",
        showgrid=False,
        zeroline=False,
        showline=False,
        showticklabels=True,
    ),
    margin=dict(
        autoexpand=False,
        l=100,
        r=100,
        t=110,
    ),
    barmode = 'group',
    legend_orientation = "h",
    showlegend=True,
    plot_bgcolor='white'
)
fig.show()
```



```
fig = px.line(contracts_comp, x="dayofyear", y="is_erc20", color = "year", hover_name= "dow
fig.update_layout(
    xaxis=dict(
        title_text = "Day Of Year",
        showline=True,
        showgrid=False,
        showticklabels=True,
        linecolor='rgb(204, 204, 204)',
        linewidth=2,
        ticks='outside',
        tickfont=dict(
            family='Arial',
            size=12,
            color='rgb(82, 82, 82)'),
    ),
    yaxis=dict(
        title_text = "ERC20 Contract Count",
        showgrid=False,
        zeroline=False,
        ticks='outside'
    )
)
```

```
snowline=False,
showticklabels=True,
),
margin=dict(
    autoexpand=False,
    l=100,
    r=100,
    t=110,
),
barmode = 'group',
legend_orientation = "h",
showlegend=True,
plot_bgcolor='white'
)
fig.show()
```



```
fig = px.line(contracts_comp, x="dayofyear", y="is_erc721", color = "year", hover_name= "do
fig.update_layout(
    xaxis=dict(
```

```
title_text = "Day Of Year",
showline=True,
showgrid=False,
showticklabels=True,
linecolor='rgb(204, 204, 204)',
linewidth=2,
ticks='outside',
tickfont=dict(
    family='Arial',
    size=12,
    color='rgb(82, 82, 82)',
),
),
yaxis=dict(
    title_text = "ERC721 Contract Count",
    showgrid=False,
    zeroline=False,
    showline=False,
    showticklabels=True,
),
margin=dict(
    autoexpand=False,
    l=100,
    r=100,
    t=110,
),
barmode = 'group',
legend_orientation = "h",
showlegend=True,
plot_bgcolor='white'
)
fig.show()
```



```
labels = np.array(contracts_comp['dow'])
feature = contracts_comp.drop('date', axis = 1)
feature_1 = feature.drop('dayofyear', axis = 1)
features = feature_1.drop('year', axis = 1)
features = features.drop('dow', axis = 1)
rf = RandomForestClassifier(n_estimators = 1000, random_state = 42)
rf.fit(features, labels)
predictions = rf.predict(features)
contracts_comp['predicted'] = predictions

contracts_comp.head()
```

↳

```
contracts_comp.count()
```

↳

```
contracts_comp[['dow']].count()
```



```
contracts_comp.groupby('dow').count()[['predicted']]
```



```
train_features, test_features, train_labels, test_labels = \
train_test_split(features, labels, test_size = 0.25, random_state = 42)
print("Size of the training data", train_features.shape)
print("Size of the test data", test_features.shape)
print("Size of the training labels", train_labels.shape)
print("Size of the test labels", test_labels.shape)
```



```
rf = RandomForestClassifier(n_estimators = 1000, random_state = 42)
rf.fit(train_features, train_labels)
predictions = rf.predict(test_features)
print(predictions)
print(test_labels)
```



```
conf_mat = pd.crosstab(predictions, test_labels, rownames=['Actual DOW'], \
                        colnames=['Predicted DOW'])
conf_mat
```



```
rf.feature_importances_
```

↳

```
list(zip(train_features, rf.feature_importances_))
```

↳

```
feature_list = list(features.columns.values)
unique_pos = contracts_comp['dow'].unique()
target_names = list(unique_pos)
estimator = rf.estimators_[0]
from sklearn.tree import export_graphviz
# Export as dot file
tree = export_graphviz(estimator, out_file='tree.dot',
                       feature_names = feature_list,
                       class_names = target_names,
                       rounded = True, proportion = False,
                       precision = 2, filled = True)

##
# Convert to png using system command (requires Graphviz)
from subprocess import call
call(['dot', '-Tpng', 'tree.dot', '-o', 'tree.png', '-Gdpi=600'])

# Display in jupyter notebook
from IPython.display import Image
Image(filename = 'tree.png')
```

↳

## ▼ *Transaction*

I ran into a problem here because the transactions table was so huge that it took too long to run and I had to break the table up and run it by week.

```
trans_18 = client.query(''  
    SELECT DATE(block_timestamp) as date, block_number, from_address, to_address, value, gas_pr  
    https://colab.research.google.com/drive/1zywqAw6uAJfxgksc4zMtV02mOXeNit0B#scrollTo=wrKLPtIUKWkf&printMode=true  
    23/33
```

```

FROM `bigquery-public-data.crypto Ethereum.transactions`
WHERE DATE(block_timestamp) >= "2018-10-01" AND DATE(block_timestamp) < "2018-10-7"').to_

trans_19 = client.query('''
SELECT DATE(block_timestamp) as date, block_number, from_address, to_address, value, gas_pr
FROM `bigquery-public-data.crypto Ethereum.transactions`
WHERE DATE(block_timestamp) >= "2019-10-01" AND DATE(block_timestamp) < "2019-10-7"').to_

trans_18_2 = client.query('''
SELECT DATE(block_timestamp) as date, block_number, from_address, to_address, value, gas_pr
FROM `bigquery-public-data.crypto Ethereum.transactions`
WHERE DATE(block_timestamp) >= "2018-10-08" AND DATE(block_timestamp) < "2018-10-18"').to

trans_19_2 = client.query('''
SELECT DATE(block_timestamp) as date, block_number, from_address, to_address, value, gas_pr
FROM `bigquery-public-data.crypto Ethereum.transactions`
WHERE DATE(block_timestamp) >= "2019-10-08" AND DATE(block_timestamp) < "2019-10-18"').to

trans_18_3 = client.query('''
SELECT DATE(block_timestamp) as date, block_number, from_address, to_address, value, gas_pr
FROM `bigquery-public-data.crypto Ethereum.transactions`
WHERE DATE(block_timestamp) >= "2018-10-19" AND DATE(block_timestamp) < "2018-10-31"').to

trans_19_3 = client.query('''
SELECT DATE(block_timestamp) as date, block_number, from_address, to_address, value, gas_pr
FROM `bigquery-public-data.crypto Ethereum.transactions`
WHERE DATE(block_timestamp) >= "2019-10-19" AND DATE(block_timestamp) < "2019-10-31"').to

frames = [trans_18, trans_18_2, trans_18_3, trans_19, trans_19_2, trans_19_3]
trans = pd.concat(frames)

trans_num = trans.groupby(by='date', as_index=False, sort = 'True').agg({'block_number': pd.
trans_from = trans.groupby(by='date', as_index=False, sort = 'True').agg({'from_address': p
trans_to = trans.groupby(by='date', as_index=False, sort = 'True').agg({'to_address': pd.Se
trans_value = trans.groupby(by='date', as_index=False, sort = 'True').agg({'value': pd.Seri
trans_price = trans.groupby(by='date', as_index=False, sort = 'True').agg({'gas_price': pd.
trans_status = trans.groupby(by='date', as_index=False, sort = 'True').agg({'receipt_status
trans_num['date'] = pd.to_datetime(trans_num['date'], errors='coerce')
trans_num['dayofyear'] = trans_num['date'].dt.dayofyear
trans_num['year'] = pd.DatetimeIndex(trans_num['date']).year
trans_num['dow'] = trans_num['date'].dt.day_name()
trans_from = trans_from[['from_address']]
trans_to = trans_to[['to_address']]
trans_value = trans_value[['value']]
trans_price = trans_price[['gas_price']]
trans_status = trans_status[['receipt_status']]
trans_comp = pd.concat([trans_num, trans_to, trans_from, trans_value, trans_price, trans_st

trans_comp.head()

```



```
fig = px.line(trans_comp, x="dayofyear", y="block_number", color = "year", hover_name= "dow
fig.update_layout(
    xaxis=dict(
        title_text = "Day Of Year",
        showline=True,
        showgrid=False,
        showticklabels=True,
        linecolor='rgb(204, 204, 204)',
        linewidth=2,
        ticks='outside',
        tickfont=dict(
            family='Arial',
            size=12,
            color='rgb(82, 82, 82)'),
    ),
    yaxis=dict(
        title_text = "Block Count",
        showgrid=False,
        zeroline=False,
        showline=False,
        showticklabels=True,
    ),
    margin=dict(
        autoexpand=False,
        l=100,
        r=100,
        t=110,
    ),
    barmode = 'group',
    legend_orientation = "h",
    showlegend=True,
    plot_bgcolor='white'
)
fig.show()
```



```
fig = px.line(trans_comp, x="dayofyear", y="from_address", color = "year", hover_name= "dow
fig.update_layout(
    xaxis=dict(
        title_text = "Day Of Year",
        showline=True,
        showgrid=False,
        showticklabels=True,
        linecolor='rgb(204, 204, 204)',
        linewidth=2,
        ticks='outside',
        tickfont=dict(
            family='Arial',
            size=12,
            color='rgb(82, 82, 82)'),
    ),
    yaxis=dict(
        title_text = "Sender Address Count",
        showgrid=False,
        zeroline=False,
        showline=False,
        showticklabels=True,
    ),
)
----- /
```

```
margin=dict(
    autoexpand=False,
    l=100,
    r=100,
    t=110,
),
barmode = 'group',
legend_orientation = "h",
showlegend=True,
plot_bgcolor='white'
)
fig.show()
```



```
fig = px.line(trans_comp, x="dayofyear", y="to_address", color = "year", hover_name= "dow",
fig.update_layout(
    xaxis=dict(
        title_text = "Day Of Year",
        showline=True,
        showgrid=False,
```

```
showticklabels=True,  
linecolor='rgb(204, 204, 204)',  
linewidth=2,  
ticks='outside',  
tickfont=dict(  
    family='Arial',  
    size=12,  
    color='rgb(82, 82, 82)',  
,  
,  
yaxis=dict(  
    title_text = "Receiver Address Count",  
    showgrid=False,  
    zeroline=False,  
    showline=False,  
    showticklabels=True,  
,  
margin=dict(  
    autoexpand=False,  
    l=100,  
    r=100,  
    t=110,  
,  
    barmode = 'group',  
    legend_orientation = "h",  
    showlegend=True,  
    plot_bgcolor='white'  
)  
fig.show()
```



```
fig = px.line(trans_comp, x="dayofyear", y="value", color = "year", hover_name= "dow", title="Average Value by Day of Year")
fig.update_layout(
    xaxis=dict(
        title_text = "Day Of Year",
        showline=True,
        showgrid=False,
        showticklabels=True,
        linecolor='rgb(204, 204, 204)',
        linewidth=2,
        ticks='outside',
        tickfont=dict(
            family='Arial',
            size=12,
            color='rgb(82, 82, 82)'
        ),
    ),
    yaxis=dict(
        title_text = "Average Value",
        showgrid=False,
        zeroline=False,
        showline=False,
        showticklabels=True,
    ),
    margin=dict(
        autoexpand=False,
        l=100,
        r=100,
        t=110,
    ),
    barmode = 'group',
    legend_orientation = "h",
    showlegend=True,
    plot_bgcolor='white'
)
fig.show()
```



```
fig = px.line(trans_comp, x="dayofyear", y="gas_price", color = "year", hover_name= "dow",
fig.update_layout(
    xaxis=dict(
        title_text = "Day Of Year",
        showline=True,
        showgrid=False,
        showticklabels=True,
        linecolor='rgb(204, 204, 204)',
        linewidth=2,
        ticks='outside',
        tickfont=dict(
            family='Arial',
            size=12,
            color='rgb(82, 82, 82)'
        ),
    ),
    yaxis=dict(
        title_text = "Average Gas Price",
        showgrid=False,
        zeroline=False,
        showline=False,
        showticklabels=True
    )
).
```

```
```
margin=dict(
    autoexpand=False,
    l=100,
    r=100,
    t=110,
),
barmode = 'group',
legend_orientation = "h",
showlegend=True,
plot_bgcolor='white'
)
fig.show()
```



```
fig = px.line(trans_comp, x="dayofyear", y="receipt_status", color = "year", hover_name= "d
fig.update_layout(
    xaxis=dict(
        title_text = "Day Of Year",
        showline=True,
```

```
showgrid=False,
showticklabels=True,
linecolor='rgb(204, 204, 204)',
linewidth=2,
ticks='outside',
tickfont=dict(
    family='Arial',
    size=12,
    color='rgb(82, 82, 82)',
),
),
yaxis=dict(
    title_text = "Failure Count",
    showgrid=False,
    zeroline=False,
    showline=False,
    showticklabels=True,
),
margin=dict(
    autoexpand=False,
    l=100,
    r=100,
    t=110,
),
barmode = 'group',
legend_orientation = "h",
showlegend=True,
plot_bgcolor='white'
)
fig.show()
```



► 2. Miners:

↳ 6 cells hidden

► Contracts Table

↳ 12 cells hidden

► Transaction Table

↳ 18 cells hidden