Installing Packages & Environment Setup

```
!pip install pandas-gbq --quiet
!pip install google-cloud-bigquery pandas
!pip install --quiet google-cloud-bigquery
from google.colab import auth
auth.authenticate_user()
import pandas as pd
from pandas.io import gbq
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from google.cloud import bigquery
```

Show hidden output

EXECUTIVE SUMMARY

```
def executive_summary():
    print("Executive Summary")
    print("------------------")
    print("This project explores the Bitcoin Cash blockchain by analyzing block-level data using BigQuery.")
    print("Our primary objective was to build a predictive model that determines whether a block exceeds")
    print("500KB in size based on metadata such as block version, number, and time since the previous block.")
    print()
    print("After cleaning and preparing 5,000 rows of blockchain data, we trained a Decision Tree Classifier.")
    print("The model achieved strong performance with an overall test accuracy of 91%.")
    print()
    print("Key results:")
    print("- Precision (for large blocks): 0.79")
    print("- Recall (for large blocks): 0.80")
    print("- F1-score (for large blocks): 0.79")
    print()
    print("The model demonstrates solid predictive power and highlights potential for deeper analysis")
    print("on blockchain behavior using machine learning techniques. Future improvements could include")
    print("tuning model hyperparameters, testing ensemble models, or analyzing trends across time windows.")

executive_summary()
```

```
Executive Summary
------------------
This project explores the Bitcoin Cash blockchain by analyzing block-level data using BigQuery.
Our primary objective was to build a predictive model that determines whether a block exceeds
500KB in size based on metadata such as block version, number, and time since the previous block.

After cleaning and preparing 5,000 rows of blockchain data, we trained a Decision Tree Classifier.
The model achieved strong performance with an overall test accuracy of 91%.

Key results:
- Precision (for large blocks): 0.79
- Recall (for large blocks): 0.80
- F1-score (for large blocks): 0.79

The model demonstrates solid predictive power and highlights potential for deeper analysis
on blockchain behavior using machine learning techniques. Future improvements could include
tuning model hyperparameters, testing ensemble models, or analyzing trends across time windows.
```

Project Connection

```
project_id = 'proven-wavelet-457219-u4'
client = bigquery.Client(project = project_id)
```

Dataset Description & Preview

```
def describe_dataset():
    print("Dataset Description")
    print("--------------------")
    print("The dataset used in this project is sourced from BigQuery’s public dataset:")
    print("`bigquery-public-data.crypto_bitcoin_cash.blocks`.")
    print()
    print("Each row in the dataset represents a block on the Bitcoin Cash blockchain.")
```

```
    print("The following variables were selected for modeling:")
    print("- size: The size of the block in bytes (used to create the target variable).")
    print("- version: The block version number.")
    print("- number: The block height (its position in the chain).")
    print("- nonce: A value miners vary to find a valid hash (excluded from final model).")
    print("- time_since_last_block: Time difference (in seconds) between this and the previous block.")
    print()
    print("The target variable `label` was engineered as a binary indicator:")
    print("- 1 if the block size > 500,000 bytes")
    print("- 0 otherwise")
    print()
    print("After filtering and cleaning, the final dataset contains 5,000 rows and 4 predictor features.")
    print("This dataset was suitable for a binary classification task.")

describe_dataset()
```

```
Dataset Description
--------------------
The dataset used in this project is sourced from BigQuery's public dataset:
`bigquery-public-data.crypto_bitcoin_cash.blocks`.

Each row in the dataset represents a block on the Bitcoin Cash blockchain.
The following variables were selected for modeling:
- size: The size of the block in bytes (used to create the target variable).
- version: The block version number.
- number: The block height (its position in the chain).
- nonce: A value miners vary to find a valid hash (excluded from final model).
- time_since_last_block: Time difference (in seconds) between this and the previous block.

The target variable `label` was engineered as a binary indicator:
- 1 if the block size > 500,000 bytes
- 0 otherwise

After filtering and cleaning, the final dataset contains 5,000 rows and 4 predictor features.
This dataset was suitable for a binary classification task.
```

```
query = """
SELECT *
FROM `bigquery-public-data.crypto_bitcoin_cash.blocks`
LIMIT 10
"""

result = client.query(query).result().to_dataframe()
result.head()
```

	hash	size	stripped_size	weight	number	version	
0	000000000ae8b7a30797a0514d0b7baa3b52f9b4b8b8ce...	372	<NA>	<NA>	63537	1	609e99894fe50491d3b08877a56d1aaf2
1	000000000be41347e44e318bfc5f4a22d0a63a10121704...	488	<NA>	<NA>	63192	1	ba96e1c4b6f51cdbd48bb3492361d03bf
2	0000000009f53a29fa9bc1ad519f28324957605bc669cc...	978	<NA>	<NA>	63381	1	e4e57b8a3b92f58dae1aa88cc0e8c752bc
3	00000000066c4809ecab54e2fb49e9be576bc1ba0e379f...	643	<NA>	<NA>	63523	1	796de6f1e005be2fa294f75f9fc48c135
4	0000000006b3b3b56f075c6e6754181630e471868cbe0f...	489	<NA>	<NA>	63120	1	882f2cee24bb2f1b8ceb2d329a41744d

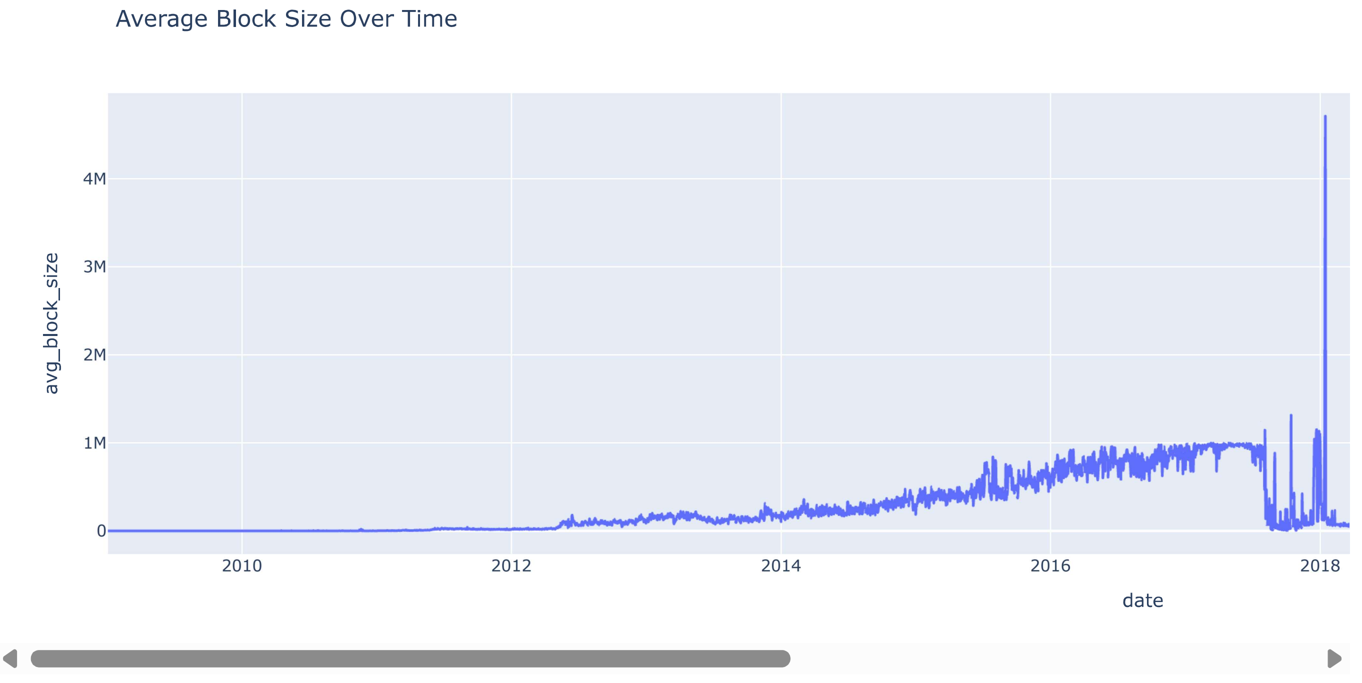
Next steps: Generate code with result | View recommended plots | New interactive sheet

EDA RESULTS and VISUALS

Query 1: Block sizes over time

```
query1 = """
SELECT
  DATE(timestamp) AS date,
  AVG(size) AS avg_block_size
FROM `bigquery-public-data.crypto_bitcoin_cash.blocks`
GROUP BY date
ORDER BY date
"""
df1 = client.query(query1).result().to_dataframe()
```

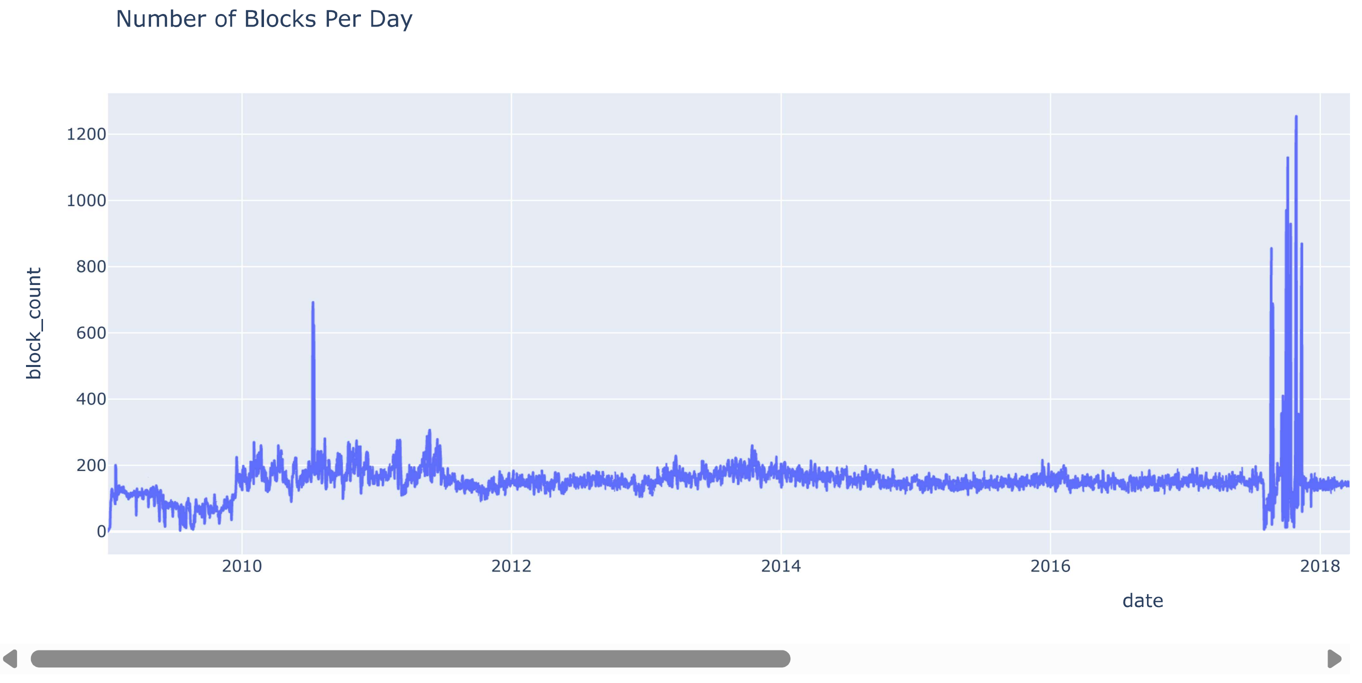
```
px.line(df1, x='date', y='avg_block_size', title='Average Block Size Over Time').show()
```



Query 2: Number of blocks per day

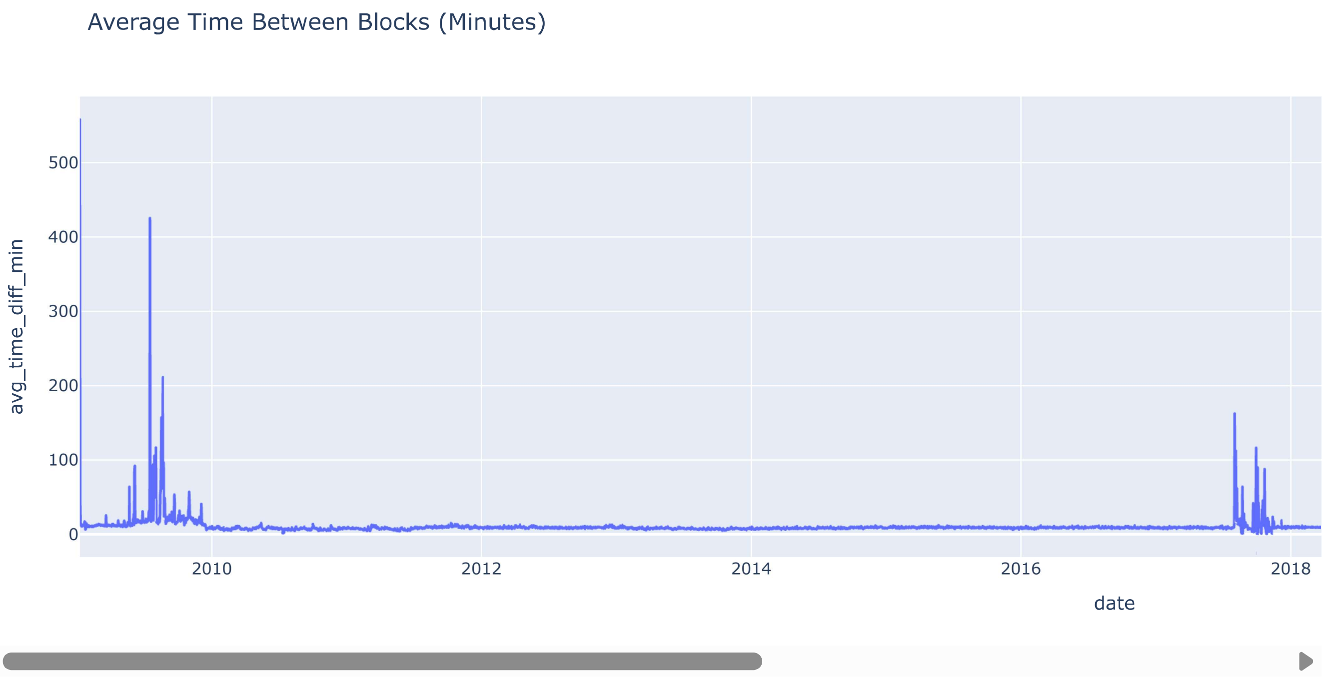
```
query2 = """
SELECT
  DATE(timestamp) AS date,
  COUNT(*) AS block_count
FROM `bigquery-public-data.crypto_bitcoin_cash.blocks`
GROUP BY date
ORDER BY date
"""
df2 = client.query(query2).result().to_dataframe()
```

```
px.line(df2, x='date', y='block_count', title='Number of Blocks Per Day').show()
```



Query 3: Average time between blocks

```
query3 = """
WITH block_times AS (
  SELECT
    timestamp,
    TIMESTAMP_DIFF(timestamp, LAG(timestamp) OVER (ORDER BY timestamp), MINUTE) AS time_diff_min
  FROM
    `bigquery-public-data.crypto_bitcoin_cash.blocks`
)

SELECT
  DATE(timestamp) AS date,
  AVG(time_diff_min) AS avg_time_diff_min
FROM block_times
WHERE time_diff_min IS NOT NULL
GROUP BY date
ORDER BY date
"""
df3 = client.query(query3).result().to_dataframe()


px.line(df3, x='date', y='avg_time_diff_min', title='Average Time Between Blocks (Minutes)').show()
```



Predictive Modeling

Query and Prepare the Data

```
query = """
WITH block_data AS (
  SELECT
    size,
    weight,
    version,
    number,
    nonce,
    TIMESTAMP_DIFF(timestamp, LAG(timestamp) OVER (ORDER BY timestamp), SECOND) AS time_since_last_block
  FROM `bigquery-public-data.crypto_bitcoin_cash.blocks`
)
SELECT *
FROM block_data
WHERE time_since_last_block IS NOT NULL
  AND size IS NOT NULL

LIMIT 5000
"""
df = client.query(query).result().to_dataframe()
df = df.drop(columns=['weight'])
df = df.dropna()
df['label'] = (df['size'] > 500000).astype(int)
print(df.shape)
print(df['label'].value_counts())
```

```
(5000, 6)
label
0    3909
1    1091
Name: count, dtype: int64
```

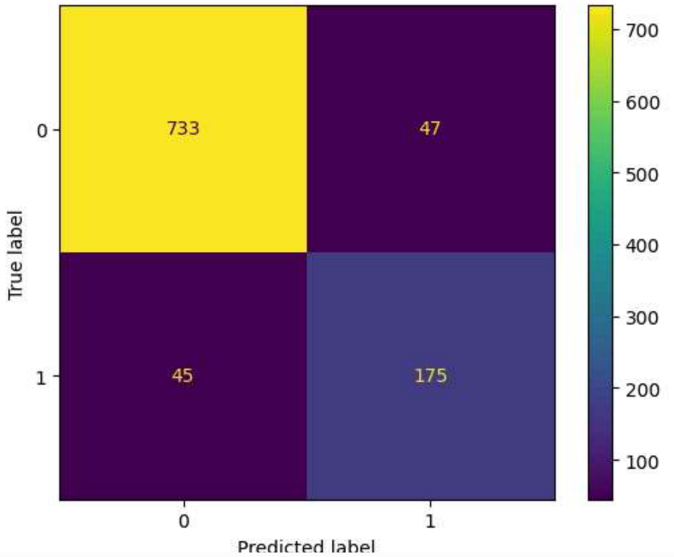
Train/Test Split and Feature Prep

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Features and target
X = df[[ 'version', 'number',  'time_since_last_block']]
y = df['label']

# Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scale features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Train & Evaluate a Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay

# Initialize and train the model
tree_model = DecisionTreeClassifier(random_state=42)
tree_model.fit(X_train_scaled, y_train)

# Predict
y_pred = tree_model.predict(X_test_scaled)

# Evaluate
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

# Optional: Visualize confusion matrix
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
```

```
[[733  47]
 [ 45 175]]
              precision    recall  f1-score   support

           0       0.94      0.94      0.94       780
           1       0.79      0.80      0.79       220

    accuracy                           0.91      1000
   macro avg       0.87      0.87      0.87      1000
weighted avg       0.91      0.91      0.91      1000

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7c35f21d6590>
```



```
def evaluate_model_performance():
    print("Model Evaluation Summary")
    print("-------------------------")
    print("We trained a Decision Tree Classifier to predict whether a block size exceeds 500KB.")
    print("The model achieved an accuracy of approximately 91% on the test set.")
    print()
    print("Key metrics:")
    print("- Precision (Class 1): 0.79 — Of all blocks predicted as large, 79% were actually large.")
    print("- Recall (Class 1): 0.80 — The model correctly identified 80% of actual large blocks.")
    print("- F1-score (Class 1): 0.79 — Balanced performance on precision and recall for large blocks.")
    print()
    print("The confusion matrix shows:")
    print("- True Negatives: 733")
    print("- False Positives: 47")
    print("- False Negatives: 45")
    print("- True Positives: 175")
    print()
    print("The model performs well overall, with stronger accuracy on the majority class.")
    print("It can be improved by trying techniques like hyperparameter tuning or using ensemble models.")

evaluate_model_performance()
```

```
Model Evaluation Summary
-------------------------
We trained a Decision Tree Classifier to predict whether a block size exceeds 500KB.
The model achieved an accuracy of approximately 91% on the test set.

Key metrics:
- Precision (Class 1): 0.79 — Of all blocks predicted as large, 79% were actually large.
- Recall (Class 1): 0.80 — The model correctly identified 80% of actual large blocks.
- F1-score (Class 1): 0.79 — Balanced performance on precision and recall for large blocks.

The confusion matrix shows:
- True Negatives: 733
- False Positives: 47
- False Negatives: 45
- True Positives: 175

The model performs well overall, with stronger accuracy on the majority class.
It can be improved by trying techniques like hyperparameter tuning or using ensemble models.
```

Managerial insights and takeaways

```
def managerial_takeaways():
    print("Managerial Insights and Takeaways")
    print("---------------------------------")
    print("1. **Blockchain block size is predictable using metadata:**")
    print("   Variables like block version, position (number), and time between blocks")
    print("   offer meaningful signals that can help anticipate whether a block will be large.")
    print()
    print("2. **Machine learning can effectively support blockchain analysis:**")
    print("   The decision tree model achieved 91% accuracy, suggesting that predictive models")
    print("   can be used for monitoring, optimization, or anomaly detection in blockchain operations.")
    print()
    print("3. **Operational planning opportunities for network scalability:**")
    print("   Knowing which blocks are likely to be large may help miners or network operators")
    print("   better manage bandwidth, node performance, and transaction prioritization.")
    print()
    print("4. **Model interpretability supports decision-making:**")
    print("   The decision tree model is transparent, making it easier to explain to stakeholders")
    print("   and adapt into rule-based systems or dashboards.")
    print()
    print("5. **Data from public sources like BigQuery can power real insights:**")
    print("   This project demonstrates how publicly available blockchain data can be")
    print("   leveraged for business intelligence and innovation.")

managerial_takeaways()
```

```
Managerial Insights and Takeaways
---------------------------------
1. **Blockchain block size is predictable using metadata:**
   Variables like block version, position (number), and time between blocks
   offer meaningful signals that can help anticipate whether a block will be large.

2. **Machine learning can effectively support blockchain analysis:**
   The decision tree model achieved 91% accuracy, suggesting that predictive models
   can be used for monitoring, optimization, or anomaly detection in blockchain operations.

3. **Operational planning opportunities for network scalability:**
   Knowing which blocks are likely to be large may help miners or network operators
```

```
   better manage bandwidth, node performance, and transaction prioritization.

4. **Model interpretability supports decision-making:**
   The decision tree model is transparent, making it easier to explain to stakeholders
   and adapt into rule-based systems or dashboards.

5. **Data from public sources like BigQuery can power real insights:**
   This project demonstrates how publicly available blockchain data can be
   leveraged for business intelligence and innovation.
```