* **Introduction**
  + Bitcoin is a decentralized cryptocurrency, created in 2009 by an anonymous inventor under the pseudonym Satoshi Nakamoto.
  + It does not have a central bank behind it that distributes new currency but relies on a network of nodes, i.e., PCs, that manage it in a distributed, peer-to-peer mode; and on the use of strong cryptography to validate and secure transactions.
  + Transactions can be made through the Internet to anyone with a "bitcoin address"
  + Bitcoin's value is determined by the market and the number of people using it.
  + Its blockchain, or public ledger of transactions, is constantly updated and validated by nodes in the network.
  + The cryptocurrency Bitcoin has attracted the attention of many people in recent years, however, it's price fluctuation can be extremely unpredictable, which makes it difficult to predict when the right time to buy or sell this digital currency will be.
  + In this context, predicting Bitcoin prices can be a competitive advantage for investors and traders, as it could allow them to make informed decisions on the right time to enter or exit the market.
  + In this project, I will analyze some machine learning techniques to understand, through the processing of historical data, how accurately the price of Bitcoin can be predicted and whether this can provide added value to cryptocurrency investors and traders.
* **Goal**
  + Is it possible to make predictions about the price of Bitcoin using machine learning methods in combination with the price information and technical characteristics of its blockchain?
* **Dataset**
  + I chose to collect data on the Bitcoin blockchain using the API of the website Blockchain.org and the price information from two famous exchanges, Binance and Kraken.
  + They retrieved the most relevant information from the last four years to the present day (a period for which there were moments of high volatility but also a lot of price lateralization).
  + The procedure has been automated in such a way that the same period is always considered
  + The features taken under consideration were divided into several categories:
    - OHLCV: stands for “Open, High, Low, Close and Volume” and it's a list of the five types of data that are most common in financial analysis regarding price
    - Currency statistics: describes its price trend (e.g. market price, number of bitcoins in circulation...)
    - Block details: describes the technical characteristics of its blockchain (e.g. block size, number of transactions...)
    - Mining information: describes the characteristics of the consensus mode “Pow” (e.g. miners revenue, difficulty...)
    - Network activity: describes the actual use of Bitcoin as a method of exchange of value (e.g. number of transactions made, cost per transaction...)
* **Project pipeline**
  + The project is structured like this:
    - Data crawling / Feature engineering: Bitcoin data retrieval via APIs call and manipulation, visualization and feature extraction
    - Models’ train / validation: performed with hyperparameter tuning and cross validation based on different methods of splitting the dataset
    - Final scores: testing the final models and compare the results
  + The project was carried out with Apache Spark (but during feature engineering I converted the Spark dataframe to a Pandas one to make some plots)
* **1. Data crawling / Feature engineering**
  + After obtaining the features regarding the technical data of the blockchain and the price of Bitcoin by contacting the APIs of Blockchain.org and the two exchanges, other features are added:
    - next-market-price: represents the price of Bitcoin for the next day (this will be the target variable on which to make predictions).
    - sma-x-days: indicators that calculate the average price over a specified number of days (5, 7, 10, 20, 50 and 100 days in our case). They are commonly used by traders to identify trends and potential buy or sell signals.
  + All these features will be divided into two distinct groups:
    - Base features: contains all the Currency Statistics features
    - Base and additional features: contains the Base features plus the additional features divided based on their correlation value with the price:
      * If >= 0.6, then they will be considered most correlated.
      * If < 0.6, then they will be considered least correlated.
  + The strategy for the model's train / validation phase will be:
    - Train / validate models with base features
    - See if by adding the additional most and least correlated features to them the performance improves
  + The whole dataset will be splitted into two sets:
    - Train / Validation set: will be used to train the models and validate the performances.
    - Test set: will be used to perform price prediction on never-before-seen data (the last 3 months of the original dataset will be used).
* **2. Models train / validation**
  + During this phase the dataset will be splitted according to different splitting method (in order to figure out which one works best for our problem):
    - Block time series splits: involves dividing the time series into blocks of equal length, and then using each block as a separate fold for cross-validation.
    - Walk forward time series splits: involves using a sliding window approach to create the training and validation sets for each fold. The model is trained on a fixed window of historical data, and then validated on the next observation in the time series. This process is repeated for each subsequent observation, with the window sliding forward one step at a time.
    - Single time series split involves dividing the time series considering as validation set a narrow period of time and as train set everything that happened before this period, in such a way as to best benefit from the trend in the short term.
  + Several types of regression algorithms will be used to see their differences and how they perform in the various stages of training / validation and testing, including:
    - Linear Regression
    - Generalized Linear Regression
    - Random Forest Regressor
    - Gradient Boosting Tree Regressor
  + Different types of metrics will be used to get a complete picture of the performance of the various models, including:
    - RMSE (Root Mean Squared Error)
    - MSE (Mean Squared Error)
    - MAE (Mean Absolute Error)
    - MAPE (Mean Absolute Percentage Error)
    - R2 (R-squared)
    - Adjusted R2
  + Since predicting the price accurately is very difficult, I also saw how good the models are at predicting whether the price will go up or down in this way:
    - For each prediction let's consider the actual market-price, next-market-price and our predicted next-market-price (prediction).
    - I compute whether the current prediction is correct (1) or not (0)
    - After that I count the number of correct prediction
    - Finally I compute the percentage of accuracy of the model
  + Concern the train / validation pipeline, it is structured like this:
    - Default without normalization: make predictions using the base model
    - Default with normalization: like the previous one but features are normalized
  + Then the features that gave on average the most satisfactory results (for each model) are chosen and proceeded with:
    - Hyperparameter tuning: finding the best model’s parameters to use. Since during this stage will be used the Block split / Walk forward split method of the dataset I compute a score for each parameter chosen by each split, assigning weights based on:
      * Their frequency for each split (if the same parameters are chosen from several splits, these will have greater weight)
      * The split they belong to (the closer the split is to today's date the more weight they will have)
      * Their RMSE value for each split (the lower this is, the more weight they will have)
      * Then, the overall score will be calculated by putting together these three weights for each parameter and the one with the best score will be the chosen parameter.
    - Cross Validation: validate the performance of the model with the chosen parameters (also here using Block split / Walk forward split)
    - If the final results are satisfactory, the model will be trained on the whole train / validation set and saved in order to make predictions on the test set.
* **3. Final scores**
  + After loading the trained models, the test set is divided into further mini-sets of 1 week, 15 days, 1 month and 3 months to see how the models' performance degrades as time increases. Final results are collected and compared to draw conclusions (see final results).
  + **Train / validation**
    - **Features**
      * **❓**
    - **Splitting method and models**
      * **❓**
  + **Test**
    - **Dataset splitting**
      * **❓**
* **Conclusions**
  + Concerning features
    - ❓
  + Concerning splitting methods and models
    - ❓
  + Concerning the dataset splitting
    - ❓
  + Regarding the initial question
    - ❓
  + Future developments
    - It might make sense to create a sliding window on features so that additional historical data is used
    - Using deep learning approaches such as CNNs (LSTM, ARIMA...) or implementing transformer models that exploit self-attention