* **Introduction**
  + Bitcoin is a decentralized cryptocurrency, created in 2009 by an anonymous inventor under the pseudonym Satoshi Nakamoto. I
  + t 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, prediction 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 exchange Binance and Kraken, the most relevant information was retrieved 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 made as automatic as possible so that the same periods are considered each time the entire procedure is run.
  + 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: describe its price trend (e.g. market price, number of bitcoins in circulation...)
    - Block details: describe the technical characteristics of its blockchain (e.g. block size, number of transactions...)
    - Mining information: describe the characteristics of the consensus mode “Pow” (e.g. miners revenue, difficulty...)
    - Network activity: describe 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: Bitcoin data retrieval via APIs call
    - Feature engineering: manipulation, visualization and feature extraction
    - Models’ train / validation: to train the models and evaluate them by performing hyperparameter tuning and cross validation based on different methods of splitting the dataset
    - Final scores: Test the final models and compare the results to answer the initial question
  + Project carried out with Apache Spark (but during feature engineering I converted the Spark dataframe to a Pandas one to make some plots)
* **Data crawling / Feature engineering**
  + I simply make a call to the APIs to retrieve the data, check for null values, and save the dataset to disk
  + **Adding new features**
    - I decided to add some features that could help us predict the Bitcoin price:
      * Next market price: represents the price of Bitcoin for the next day (this will be the target variable on which to make predictions)
      * Rate of change: indicator that measures the percentage of price changes over a period of time, allows investors to spot security momentum and other trends
      * Simple Moving Averages: indicators that calculate the average price over a specified number of days. They are commonly used by traders to identify trends and potential buy or sell signals
  + **Dataset spit**
    - I decided to split the dataset 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).
  + **Feature selection**
    - All the features will be divided into two distinct groups:
      * Currency features: contains currency statistics and ohlcv statistics
      * Currency and blockchain features: contains the currency features plus the blockchain features divided based on their correlation value:
      * If >= 0.5, then then they will be considered the \*\*most correlated\*\*
      * If < 0.5, then then they will be considered the \*\*least correlated\*\*
    - The strategy for will be as follows:
      * Test models with currency features
      * See if by adding the blockchain most and least correlated features to them improves the situation
* **Models train / validation**
  + Several types of regression algorithms will be used, including: Linear Regression, Generalized Linear Regression, Random Forest Regressor and Gradient Boosting Tree Regressor
    - To see their differences and how they perform in the various stages of training / validation and testing.
  + Different types of metrics will be used, including: RMSE (Root Mean Squared Error), MSE (Mean Squared Error), MAE (Mean Absolute Error), MAPE (Mean Absolute Percentage Error), R2 (R-squared) and Adjusted R2
    - To get a complete picture of the performance of the various models.
    - Since predicting the price accurately is very difficult we will see also how good the models are at predicting whether the price will go up or down.
      * We compute whether each prediction is correct (1) or not (0)
      * Then we count the number of correct prediction
      * Finally we compute the percentage of accuracy of the model
  + **Pipeline**
    - To train and validate the model I will try several approaches:
      * Default without normalization: Make predictions using the chosen 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: Researching the best parameters to use
        + Since I will use the walking forward method during this phase 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)

* + - * + Finally, the overall score will be calculated by putting together these 3 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
    - If the final results are satisfactory, the model will be trained on the whole train / validation set and saved to later make predictions on the test set.
    - For each approach the train / validation set will be split according to the chosen 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.
  + **Train / validation results**
    - **❗**
* **Models testing** 
  + **Pipeline**
    - After loading the previously trained models on the whole train / validation set, 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.
  + **Results**
    - **❗**
* **Conclusions**
  + Concerning features
    - Only the technical data features of the blockchain are not sufficient to predict the price of bitcoin
  + Concerning splitting methods
  + About models
  + Regarding the initial question
  + Future developments