

# WORK-FLOW PROGETTO

## – FASE 2 –

1. **ESTRAZIONE DEI DATI:** Usando la classe “**JSON\_Saver**”, estrai tutti i tipi di candele del Market (Bitstamp-btcusd) che vanno dal 01-01-2004 ad ora e salvale in una serie di file .json.

Nello specifico, la query che dovrà eseguire la classe **Extractor** è la seguente:

<https://api.cryptowat.ch/markets/bitstamp/btcusd/ohlcv?after=1072911600>

Alla fine di questa operazione, avremo una partizione (l'insieme dei file json) creata a partire dall'insieme iniziale (il singolo json restituito dalla query).

2. **REFINING DEI DATI:** Sfruttando la classe “**Dataset\_Maker**”, creare il Dataset a partire dal .json che contiene le candele da 12 h.
3. **VISUALIZZAZIONE DEI DATI:** Sfruttando le librerie Google Charts oppure HighCharts, costruire ,nei limiti del possibile, i seguenti Grafici per un insieme significativo di tipi di Candele (ad esempio, fare i grafici solamente per le candele di 12 h e di 6h) : Spearman, Scatter-Plot Matrix.  
Inoltre, calcolare i seguenti indici: Covarianza, Coefficiente di Pearson, Indice di Kandal, Matrice di Correlazione, Media e Varianza.

#### 4. **CLEANING DEI DATI:**

- Valutare se fare Feature Scaling, Mean Normalization o PCA dei nostri Dati, in tal caso usare le classi “**Cleaner**” e “**PCA**”.

#### 5. **LEARNING:**

- Sfruttando la classe “**Dataset\_Splitter**”, suddividi il Dataset delle Candele da 12 h in Test- Set e Training-Set.
- Creare una lista di modelli, sfruttando il metodo **build** della classe **Model\_Builder**.
- A questo punto passare la Lista dei Modelli al metodo **train\_models** della classe **Trainer**.
- Fatto ciò, chiamare il metodo **plot\_graphics** della classe **Trainer** che non farà altro che plottare tutti i Grafici delle Ipotesi (in 2D) di tutti i modelli allenati (qua si sfrutta la classe **Curve**); questo metodo plotta anche i Grafici Grado vs Errore se sono presenti ,nella lista dei Modelli Allenati, gruppi di Modelli identici con grado ascendente.
- Arrivati a questo punto, stampare tutti gli errori di J-CV e J-Train per ogni modello sfruttando il metodo **print\_models\_errors** della classe **Trainer**.
- Inoltre, salvare il miglior modello sfruttando il metodo **save\_best\_estimator** della classe “**Trainer**”.

- Se si fanno allenamenti multipli dello stesso modello cambiando il grado, registrare i cambiamenti dell'errore ,fra gli allenamenti, rispetto al grado con la classe **“Record\_Degree”**
- Se si fanno allenamenti multipli dello stesso modello cambiando lambda (qua siamo per forza nella Discesa del Gradiente), registrare i cambiamenti dell'errore ,fra gli allenamenti, rispetto a lambda con la classe **“Record\_Lambda”**
- Se si è scelta una strategia ad allenamenti multipli, e si è cambiato grado del Modello durante gli allenamenti: plottare il Grafico “Grado del Polinomio vs Errore”
- Se si è scelta una strategia ad allenamenti multipli, e si è cambiato lambda durante gli allenamenti, plottare il Grafico “Grado Lambda vs Errore”

## **6.Valutazione dei Risultati:**

- Usare il metodo **take\_the\_best\_model\_from\_last\_training** della classe **“Trainer”** per estrapolare il miglior modello.
- Fatto ciò, calcolare l'MSE sul Test-Set sfruttando il Modello Estratto al punto precedente.