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/ohlc?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 (suddiviso in 5 parti).
   - Nomina Iteativamente una porzione di Training-Set come Validation-Test
   - PER OGNI PORZIONE DI TRAINING DIVENTATA VALIDATION:
     - Sfruttando la classe **“Dataset_Splitter”** (metodo **merge),** unisci le restanti 4 parti di Training-Set in un’unica struttura, questo sarà il nostro Training-Set per questa iterazione.
     - Stilare una serie di modelli per cui si vuole fare il Learning (possono essere anche uguali per tutte le iterazioni).
     - PER OGNI MODELLO:
       - Apprendere i migliori parametri theta sfruttando la classe **“Normal_Equations”** o la classe **“Gradient_Descent”**
       - Calcolare la J-train sfruttando la classe **“Cost_Function”** se si sta usando la Discesa, altrimenti usare la classe **“MSE”**
       - Calcolare la J-CV usando la classe **“MSE”**
       - Plottare il Grafico dell’Ipotesi in 2D sfruttando la classe **“Curve”**
       - 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_Degree"**
        - 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 del Polinomio vs Errore"

    - Scegliere il Modello che ha il J-CV più basso.

- Scegliere il Modello ,fra i 5 ottenuti dal 5-Cross-Validation, che ha errore sul Validation Test più piccolo.

6.Valutazione dei Risultati:

- Calcolare l'MSE dell'Algortimo. (Sul Test-Set ovviamente).