TEMPLATE DI PROJECT WORK = 3 CFU **min 12 pagine - max 20 pagine** **da compilare e caricare in formato pdf*	
Cognome e Nome:	Panepinto Carmelo
Numero di Matricola:	0312300569
Corso di Studio: ◊ L-5 Filosofia ed Etica ◊ L-22 Scienze Motorie ✘ ◊ L-31 Informatica per le Aziende Digitali	Barrare la casella riferita al proprio corso di studio
Tema n: **Titolo del tema:**	1 La digitalizzazione dell'impresa
Traccia del PW n: **Titolo della traccia:**	1.6 Sviluppo di una dashboard in Python per l'analisi delle prestazioni aziendali nel settore primario
Titolo dell'elaborato:	Dashboard Analisi Prestazioni Aziendali - Settore Primario
PARTE PRIMA – DESCRIZIONE DEL PROCESSO	
Utilizzo delle conoscenze e abilità derivate dal percorso di studio:	Il progetto applicherà le competenze acquisite nella programmazione con linguaggio Phyton, nell'analisi dati tramite grafici appositi e nell'architettura di un applicativo informatico.
Fasi di lavoro e relativi tempi di implementazione per la predisposizione dell'elaborato:	Il project work ha richiesto una suddivisione dello sviluppo in fasi, con intervallo di tempo specifico per ogni fase, per un totale di circa 2 mesi. Le fasi possono essere riassunte in 5 parti: 1. Analisi Iniziale; 2. Sviluppo del Simulatore; 3. Sviluppo della Dashboard; 4. Fase di test; 5. Stesura del Report.

In particolare, ciascuna fase dello sviluppo e le tempistiche relative sono descritte nel seguente modo:

- **Fase 1, Analisi Iniziale** (Una settimana): Studio approfondito della traccia del PW e definizione degli obiettivi del progetto. Ideazione iniziale del progetto, con stesura di schemi scritti e diagrammi UML per avere un'immagine mentale dell'intero progetto. Scelta dell'ambiente su cui lavorare, l'output desiderato e i dati richiesti per il funzionamento del progetto;

- **Fase 2, Inizio dello sviluppo (Simulatore)** (Due settimane): Lo sviluppo è cominciato con la creazione della three folder, delle sottocartelle e la creazione e suddivisione dei vari file nelle cartelle.

  In seguito, sono partito dal simulatore: questo deve simulare i dati per una serie di parametri specifici, per cui ho importato le librerie necessarie e ho ideato uno specifico algoritmo matematico per far sì che questi dati siano verosimili e realistici. A partire da tali dati, il codice deve generare una quantità X di resa.

  In fine, ho pensato di esportare in un file CSV i dati generati, che verranno poi utilizzati dalla dashboard.

  Sono seguiti giorni di test e prove per assicurarmi che il risultato sia accettabile, pulire e commentare il codice.

- **Fase 3, Sviluppo della Dashboard** (Due settimane): Una volta accertato che i risultati del simulatore fossero idonei, ho proceduto con la creazione dell'interfaccia interattiva tramite la libreria Dash di Phyton.
  Innanzitutto ho creato, tramite le librerie numpy e pandas, il grafico principale della resa; ho poi creato i grafici di tutti gli altri parametri e inserito un menù a tendina da dove poterli selezionare per visionarli. Infine, ho inserito un grafico di confronto tra due parametri specifici con una retta di regressione per indicarne la corrispondenza.
  In seguito ho pensato di poter inserire l'intervallo di giorni desiderato per visionare i dati, per una visione più specifica e dettagliata.

  Circa 3 giorni sono stati dedicati alla fase di test e debug, alla pulizia del codice e ai commenti.

	In fine, ho creato un file css da importare nella dashbord per rendere l'interfaccia più user-friendly, colorata e strutturata. • **Fase 4, Analisi Finale e Test** (Una settimana): Una volta terminata la stesura dei codici, ho dedicato una settimana per un ultimo controllo nel codice, aggiungere altre idee e testate la validità del risultato finale. • **Fase 5, Stesura del Report** (Una settimana): L'ultima settimana è stata dedicata alla stesura del report che descrive il PW, le fasi del processo e le tecnologie utilizzate.
Risorse e strumenti impiegati:	Linguaggi di programmazione: Python. Librerie utilizzate: Pandas, NumPy, Matplotlib, Plotly, Dash. Ambiente di sviluppo: Visual Studio Code. Strumenti di versioning: Git e GitHub. Dataset: Generati tramite simulatore personalizzato. Documentazione e risorse online: Articoli accademici, documentazione ufficiale delle librerie utilizzate.
PARTE SECONDA – PREDISPOSIZIONE DELL'ELABORATO	
Obiettivi dell'elaborato/progetto/artefatto:	LINK REPOSITORY AL PROGETTO: https://github.com/Carmoz00/Project-Work L'obiettivo del progetto è la realizzazione di un sistema di monitoraggio delle prestazioni aziendali nel settore primario attraverso una dashboard interattiva. In questo caso, verrà presa in considerazione un'ipotetica azienda agricola che si occupa della produzione del grano. L'applicazione, partendo da parametri verosimili quali: • Temperatura (°C); • Umidità (%); • Precipitazioni (mm);

	• Ore di luce giornaliere (h); • Uso Fertilizzanti (Kg); • Consumo Acqua (L) Genera un determinato valore di raccolto del grano. La dashboard mostrerà tutti i grafici e le statistiche apposite, consentendo di confrontare parametri, trovare correlazioni e punti di criticità. Mostrerà inoltre un grafico di confronto tra parametri con una retta di regressione per il rapporto umidita/consumo acqua (grafico che può essere ripetuto per ciascun parametro).
Contestualizzazione:	Il settore primario, in particolare quello agricolo, necessita di una innovazione tecnologica soprattutto per la raccolta e l'analisi dei dati, i quali giocano un ruolo fondamentale per ottimizzare la produzione e ridurre gli sprechi. Conoscendo bene il settore, in quando la mia famiglia ha un'azienda agricola, so bene che aiuti può dare la tecnologia in questo ambito, soprattutto se utilizzata nel modo opportuno. Conoscere nel dettaglio dati e statistiche relative alla produzione aziendale offre una chiara visione dell'andamento dell'azienda, permette il riconoscimento di punti deboli e punti di forza e consente di risolvere eventuali criticità.
Descrizione dei principali aspetti progettuali:	Il sistema sviluppato si compone di due parti principali: 1. **Simulatore di dati**: Questo codice simula un anno di produzione, suddiviso in 365 giorni. Per ogni giorno, viene generato in maniera parzialmente random il valore dei parametri descritti precedentemente (temperatura, umidità, precipitazioni, fertilizzanti, consumo acqua, ore di luce) dai quali viene generato il valore giornaliero del raccolto. La randomizzazione dei dati segue un determinato algoritmo, con distriuzione normale e deviazione standard, per assicurare che siano dati realistici, per esempio viene considerato l'alternarsi delle stagioni, che comporterà temperature più altre in estate e più basse in inverno (ciò si applica su tutti i parametri). Inoltre, alcuni dati dipendono l'uno dall'altro, ad esempio per un minore valore di precipitazioni ci sarà un maggiore utilizzo di fertilizzanti. Il valore della resa sarà quindi dipenderà quindi da molteplici variabili al fine di essere verosimile. Infine, il codice esporterà i dati in uno specifico file CSV, che verrà poi utilizzato nella dashboard come banca dati.

Tecnologie utilizzate in questo codice:

Librerie:

- **NumPy**: Utilizzata per la generazione di numeri casuali e la manipolazione di array numerici.
- **Pandas:** Utilizzata per la gestione e l'elaborazione dei dati in tabella.

Funzioni e metodi:

- **np.arange(start, stop):** Crea un array con valori che vanno da start a stop. Utilizzato per i giorni dell'anno;
- **np.sin(x):** Calcola il seno di un valore x. Utilizzato per simulare variazioni stagionali nei dati.
- **np.random.normal(loc, scale, size):** Genera numeri casuali da una distribuzione normale con media loc e deviazione standard scale.
- **np.random.uniform(low, high, size):** Genera numeri casuali da una distribuzione uniforme tra low e high.
- **np.maximum(x, y):** Restituisce il massimo tra x e y. Utilizzato per evitare valori negativi nelle precipitazioni.
- **np.clip(array, min, max):** Limita i valori di un array a un intervallo compreso tra min e max. Utilizzato per le ore di luce e l'umidità.
- **pd.DataFrame(data):** Crea un DataFrame da un dizionario data.
- **df.iloc[start:end]:** Seleziona un sottoinsieme di righe da start a end.
- **df.to_csv(path, index=False):** Salva il DataFrame in un file CSV senza salvare gli indici delle righe.
- **pd.concat()** unisce due DataFrame lungo le colonne.

2. **Dashboard interattiva**:

   La dashboard di occupa di rappresentare i dati generati in grafici. I dati, letti dal file csv creato dal simulatore, vengono convertiti in grafici dettagliati che saranno mostrati nell'interfaccia e, tramite un menù a tendina, sarà possibile selezionare il grafico desiderato e visualizzarlo.

   Il grafico principale resta quello della resa della produzione del grano, che sarà sempre visibile. Nella parte inferiore della pagina, invece, ci sarà il grafico di confronto tra parametri.

Tecnologie utilizzate:

Librerie:

- **Dash:** Utilizzata per creare l'interfaccia web della dashboard.
- **Dash Bootstrap Components**: Utilizzata per migliorare il layout e la grafica con componenti predefiniti.
- **Pandas**: Utilizzata per leggere e manipolare il dataset dei dati simulati.
- **Plotly Express**: Utilizzata per creare i grafici.

Funzioni e metodi:

- **pd.read_csv(path**): Legge un file CSV e lo trasforma in un DataFrame Pandas.
- **dash.Dash(__name__):** Crea un'app Dash.
- **external_stylesheets**: Carica un tema Bootstrap per migliorare l'aspetto grafico.
- **html.H1("Dashboard Analisi Aziendale"):** Crea un titolo nella pagina.
- **dcc.Dropdown():** Crea un menu a tendina per selezionare i parametri da visualizzare.
- **dcc.RangeSlider():** Crea uno slider per selezionare l'intervallo di giorni.
- **html.Div(id='grafici-parametri'):** Contiene i grafici aggiornati dinamicamente.
- **dcc.Graph(id='grafico-produzione'):** Contiene il grafico della resa.
- **@app.callback(Output('grafici-parametri')):** Ogni volta che vengono scelti i grafici tramite il menù a tendina o l'intervallo dei giorni, la funzione viene eseguita.
- **df.iloc[intervallo[0]-1:intervallo[1]]:** Filtra il dataset per l'intervallo selezionato.
- **px.line(df_filtro, x='Giorno', y=p, title=f'Andamento {p}'):** Crea un grafico a linee con l'andamento del parametro selezionato.
- **px.scatter(df_filtro, x='Umidità (%)', y='Consumo Acqua Grano (litri)', title='Correlazione Umidità e Consumo Acqua', trendline='ols'):** Crea scatterplot per evidenziare la correlazione tra umidità e consumo di acqua, aggiungendo una trendline per facilitare la lettura dei dati.

Campi di applicazione:	Il progetto è in realtà applicabile in qualsiasi azienda del settore primario. Oltre ad un'azienda agricola, si potrebbe utilizzare infatti fare per un'azienda tessile, un caseificio, un'azienda manifatturiera ecc. (con i rispettivi parametri), in generale per una qualsiasi azienda che produce beni e che opera con numerose risorse e parametri.
Valutazione dei risultati (potenzialità e criticità):	L'applicativo possiede la potenzialità di innovare tecnologicamente un settore lavorativo come quello primario che ha necessità di adeguarsi ai nuovi strumenti informatici. Come già detto, può essere applicato in praticamente tutti i campi di questo settore: l'applicativo sarà in grado di trasporre qualsiasi tipo di dato, grazie alla sua scalabilità. Inoltre, grazie al linguaggio phyton e alla struttura semplice ed intuitiva del progetto, è facilmente aggiornabile, modificabile e adattabile nel contesto desiderato. Una delle criticità dell'applicativo è al momento la dipendenza da dati simulati, che possono però essere tranquillamente sostituiti da dati reali di un'azienda esistente. Inoltre, si potrebbe aggiungere la possibilità di creare un report direttamente dalla dashboard, con grafici e dati, e poterlo scaricare e stampare.