EPICODE CAPSTONE RELAZIONE FINALE – ETF ANALYSIS

Indice

OBIETTIVO 2

1. SCRAPING 4

1.1 Ottenimento dei dati per settore 4

1.2 Concatenazione Settoriale 5

1.3 Merge settoriale finale 6

2. DATA CLEANING & MANIPULATION 6

2.1 Normalizzazione DataBase & Prime Analisi 6

2.2 Grafici 8

2.2 KPI 9

3. DATA VISUALIZATION 11

3.1 POWER BI 12

CONCLUSIONI 15

OBIETTIVO

Come progetto finale del corso di Data Analyst svolto presso Epicode School, ho deciso di svolgere un'analisi di performance/ rischio di alcuni ETF (Exchange-Traded Fund), uno strumento finanziario che mi ha da sempre affascinato.

Data la tecnicità dell'argomento è stata una scelta congrua al fine di mettere in pratica diversi strumenti utilizzati durante i vari moduli del corso.

Ho diviso il progetto in 3 sezioni:

- Data Scraping: effettuata con Python
- Data Manipulation/ Analysis: effettuata con Pandas (Python) ma anche Excel e Power Query
- Data Visualization: effettuata con Power BI.

Prima di iniziare a spiegare come ho svolto le varie fasi, ecco un po' di logica di base che ho utilizzato per la creazione del mio DataBase da analizzare.

Essendo gli ETF un argomento sia incredibilmente complesso e vasto da analizzare, ho incentrato la ricerca su diversi ETF Settoriali.

La prima parte dell'analisi è stata, quindi, la suddivisione settoriale degli ETF da analizzare in 5 diversi settori:

- Settore Tecnologico
- Settore Energetico
- Settore Finanziario
- Settore Sanitario
- Settori Beni di Consumo

Per ogni settore sono stati selezionati 5 ETF differenti per avere un numero di campi e records rilevanti per una analisi approfondita dell'argomento.

Qui gli ETF selezionati per ogni settore:

1. Settore Tecnologico:

    - Technology Select sector SPDR FUND (XLK)
    - Invesco QQQ Trust (QQQ)
    - Ishare US technology ETF (IYW)
    - Vanguard Information Technology ETF (VGT)
    - First Trust Down Jones Internet Index Fund (FDN)

2. Settore Energetico:

    - Energy Select Sector SPDR Fund (XLE)
    - iShares Global Energy ETF (IXC)
    - VanEck Vectors Oil Services ETF (OIH)
    - SPDR S&P Oil & Gas Exploration & Production ETF (XOP)
    - iShares U.S. Energy ETF (IYE)

3. Settore Finanziario:

    - Financial Select Sector SPDR Fund (XLF)
    - iShares U.S. Financials ETF (IYF)
    - Invesco KBW Bank ETF (KBWB)
    - SPDR S&P Regional Banking ETF (KRE)
    - Invesco KBW Property & Casualty Insurance ETF (KBWP)

4. Settore Sanitario:

    - Health Care Sector SPDR Fund (XLV)
    - Vanguard Health Care (VHT)
    - Ishares Us. Healthcare ETF (IYH)
    - Invesco S&P500 Equal Weight Health Care ETF (RYH)
    - Ishares Global Healthcare (IXJ)

5. Beni di Consumo:

    - Consumer Discretionary Select Sector SPDR Fund (XLY)
    - Vanguard Consumer Discretionary ETF (VCR)
    - Invesco Dynamic Leisure and Entertainment ETF (PEJ)
    - ProShares Online Retail ETF (ONLN)
    - VanEck Vectors Gaming ETF (BJK)

I dati sono stati reperiti da fonti differenti tra le quali:

- Yahoo Finance
- Investing.com
- Justetf.com
- ETF.com

1. SCRAPING

Per reperire I dati, ho voluto mettere in pratica ciò che ho utilizzato nel modulo di Python e ho aggiunto qualcosa in più per fare pratica. Lo scopo di questo processo è stato quello di reperire dati grezzi per gli ETF selezionati – utilizzando la libreria yahoofinance direttamente tramite Python in modo da automatizzare il processo, invece di scaricare dati manualmente.

Il processo di Scraping è stato diviso in 3 fasi:

1.1 Ottenimento dei dati per settore

Lo script che allego, è stato utilizzato per reperire i ticker degli ETF selezionati per la categoria Technology. Il processo è stato poi replicato per ogni settore, così da avere 5 file.csv differenti. Ho inserito le date di estrazione per 5 anni (obiettivo della ricerca e con estrazione mensile).

In [1]:
```
import yfinance as yf
import pandas as pd
import glob
```

In [2]:
```
# Definisci i ticker degli ETF e i nomi desiderati per i file CSV
etf_info = [
    {"ticker": "QQQ", "csv_name": "QQQ_technology.csv"},
    {"ticker": "VGT", "csv_name": "VGT_technology.csv"},
    {"ticker": "FDN", "csv_name": "FDN_technology.csv"},
    {"ticker": "XLK", "csv_name": "XLK_technology.csv"},
    {"ticker": "IYW", "csv_name": "IYW_technology.csv"}
]
```

In [3]:
```
# Scarica i dati storici mensili degli ultimi 5 anni per ciascun ETF


for etf in etf_info:
    etf_ticker = etf["ticker"]
    etf_data = yf.download(etf_ticker, start="2018-01-01", end="2023-01-01", interval="1mo")
    csv_filename = etf["csv_name"]
    etf_data.to_csv(csv_filename)
    print(f"Dati di {etf_ticker} salvati in {csv_filename}")
```

```
[**********100%%**********]  1 of 1 completed
Dati di QQQ salvati in QQQ_technology.csv
[**********100%%**********]  1 of 1 completed
Dati di VGT salvati in VGT_technology.csv
[**********100%%**********]  1 of 1 completed
Dati di FDN salvati in FDN_technology.csv
[**********100%%**********]  1 of 1 completed
Dati di XLK salvati in XLK_technology.csv
[**********100%%**********]  1 of 1 completed
Dati di IYW salvati in IYW_technology.csv
```

1.2 Concatenazione Settoriale

Dopo avere scaricato nella mia Directory i 5 file.csv per settore, ho concatenato tutti e 5 files .csv in un unico file .csv settoriale, cosi da essere più efficace al momento delle analisi del Database.

Esempio di codice utilizzato per concatenare tutti i singoli csv settoriali in un unico file utilizzando la funzione pd.concat.

In [1]:
```
import yfinance as yf
import os
import pandas as pd
```

In [2]:
```
current_directory = os.getcwd()
```

In [3]:
```
# Lista di ETF nello stesso settore (es. Technology)
etf_data_technology = [
    {"ticker": "QQQ", "sector": "Technology"},
    {"ticker": "XLK", "sector": "Technology"},
    {"ticker": "VGT", "sector": "Technology"},
    {"ticker": "FDN", "sector": "Technology"},
    {"ticker": "IYW", "sector": "Technology"},
]
```

In [4]:
```
# Lista per conservare i DataFrame di ciascun ETF
dfs = []
```

In [5]:
```
# Itera sulla lista e scarica i dati mensili per ciascun ETF

for etf in etf_data_technology:
    ticker = etf["ticker"]
    sector = etf["sector"]

    #Scarica solo i dati storici mensili per l'ETF corrente
    data = yf.download(ticker, period='5y', interval='1mo')  # interval='1mo' per dati mensili

    # Ottenere il nome completo dello strumento finanziario
    full_name = yf.Ticker(ticker).info['longName']


    # Aggiungi le colonne del settore e del nome ETF
    data['sector'] = sector
    data['ETF Name'] = ticker
    data['Full ETF Name'] = full_name


    # Aggiungi il DataFrame all'elenco
    dfs.append(data)
```

```
[**********100%%**********]  1 of 1 completed
[**********100%%**********]  1 of 1 completed
[**********100%%**********]  1 of 1 completed
[**********100%%**********]  1 of 1 completed
[**********100%%**********]  1 of 1 completed
```

```
# Concatena tutti i DataFrame in un unico DataFrame
all_etf_data_technology = pd.concat(dfs, axis=0)
```

```
# Esporta i dati combinati in un singolo file CSV

combined_file_name = f"all_etf_technology.csv"
combined_file_path = os.path.join(current_directory, combined_file_name)
all_etf_data_technology.to_csv(combined_file_path)

print(f"I dati combinati di tutti gli ETF sono stati esportati in {combined_file_path}")
```

```
I dati combinati di tutti gli ETF sono stati esportati in C:\Users\DELL\all_etf_technology.csv
```

1.3 Merge settoriale finale

Lo step successivo, è stato inserire tutti i file settoriali completi - ottenuti dalla fase precedente, in un unico file definitivo con tutti e 25 gli ETF selezionati (denominato " dati_combined_finale.csv") – così da avere in un unico file.csv tutti gli ETF selzionati per ogni settore di riferimento.

In [15]:
```
import pandas as pd
import os
```

In [16]:
```
# Specifica il percorso della cartella contenente i tuoi file CSV

folder_path = 'C:\\Users\DELL\\ETF da concatenare'
```

In [17]:
```
# Lista vuota per contenere i DataFrame di ciascun file
dfs = []
```

In [18]:
```
# Ciclo attraverso ciascun file nella cartella

for filename in os.listdir(folder_path):
    if filename.endswith(".csv"):
        file_path = os.path.join(folder_path, filename)
        df = pd.read_csv(file_path)
        dfs.append(df)
```

In [19]:
```
# Concatena i DataFrame in un unico DataFrame
merged_df = pd.concat(dfs, axis=0, ignore_index=True)
```

In [13]:
```
# Salva il DataFrame combinato in un nuovo file CSV
merged_df.to_csv('C:\\Users\\DELL\\ETF da concatenare\\dati_combined_finale.csv', index=False)
```

In [14]:
```
print("Concatenazione completata. Il file combinato è stato salvato.")
```

```
Concatenazione completata. Il file combinato è stato salvato.
```

2. DATA CLEANING & MANIPULATION

2.1Normalizzazione DataBase & Prime Analisi

Per la normalizzazione del DB, la parte di cleaning e analisi, ho utilizzato Pandas di Python ma anche Excel & Power Query.

Al fine di capire se il DB presentasse delle anomalie in termini di data missing, blank, null o anche solo outlier, ho iniziato con una Exploratory Analysis.

- Per visualizzare le prime 5 righe del DB ho utilizzato una print(df.head())

```
In [6]: # Visualizzare le prime righe del DataFrame

        print(df.head())
```

```
         Date        Open        High         Low       Close   Adj Close  \
0  2018-12-01  110.209999  110.599998   91.730003   99.010002   94.173820
1  2019-01-01   97.320000  109.150002   96.790001  108.779999  103.872147
2  2019-02-01  107.849998  111.910004  106.360001  110.209999  105.237625
3  2019-03-01  111.110001  114.059998  107.660004  113.849998  108.713387
4  2019-04-01  114.769997  120.900002  114.410004  120.070000  115.044006

      Volume            sector ETF Name  \
0  184699100  Consumer Service      XLY
1  114456600  Consumer Service      XLY
2   82301900  Consumer Service      XLY
3  126494800  Consumer Service      XLY
4   72082500  Consumer Service      XLY

                                  Full ETF Name
0  Consumer Discretionary Select Sector SPDR Fund
1  Consumer Discretionary Select Sector SPDR Fund
2  Consumer Discretionary Select Sector SPDR Fund
3  Consumer Discretionary Select Sector SPDR Fund
4  Consumer Discretionary Select Sector SPDR Fund
```

- Per verificare la presenza o meno di missing data ho utilizzato: print (df.info ())

```
n [9]: # Info riguardo il DF - tipi di dati, eventuali valori null e utilizzo di memoria
       print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1495 entries, 0 to 1494
Data columns (total 10 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Date           1495 non-null   object
 1   Open           1495 non-null   float64
 2   High           1495 non-null   float64
 3   Low            1495 non-null   float64
 4   Close          1495 non-null   float64
 5   Adj Close      1495 non-null   float64
 6   Volume         1495 non-null   int64
 7   sector         1495 non-null   object
 8   ETF Name       1495 non-null   object
 9   Full ETF Name  1495 non-null   object
dtypes: float64(5), int64(1), object(4)
memory usage: 116.9+ KB
None
```

- Per un ulteriore verifica, mi è sembrato giusto eseguire una print(df.isnull().sum())

```
|: # Verifica di eventuali valori mancanti

   print(df.isnull().sum())
```

```
Date            0
Open            0
High            0
Low             0
Close           0
Adj Close       0
Volume          0
sector          0
ETF Name        0
Full ETF Name   0
dtype: int64
```

- Una volta verificato l'assenza di missing data nel DB, ho iniziato con delle statistiche descrittive per esempio media, min,max, standard deviation.

```
pd.set_option('display.float_format', '{:.2f}'.format) # Questa riga di codice mi serve per visualizzare due decimali

print(df.describe())
```

```
         Open    High     Low   Close  Adj Close        Volume
count 1495.00 1495.00 1495.00 1495.00    1495.00       1495.00
mean   125.76  132.60  118.71  126.21     122.98  151901153.60
std     97.20  101.71   92.15   97.61      96.89  326597960.70
min     14.85   16.89   12.13   15.31      13.26      42400.00
25%     47.42   50.68   43.87   47.43      44.94    2608500.00
50%     84.28   88.78   80.61   84.48      82.73   11660100.00
75%    187.17  196.90  177.03  188.28     182.35  138789500.00
max    458.73  467.06  432.00  460.37     460.37 3049399400.00
```

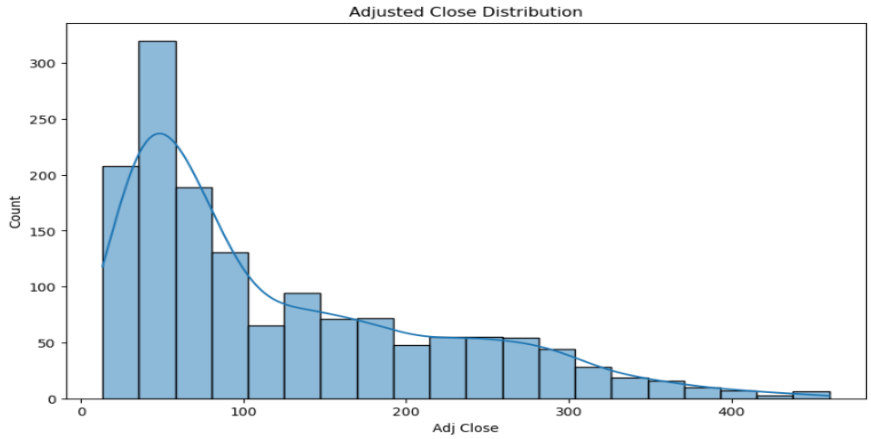
2.2 Grafici

Una volta terminata la parte di data validation e statistica descrittiva, ho fatto dei grafici con Python e iniziato a calcolare delle KPI - utili allo studio degli ETF selezionati. Per i grafici ho utilizzato Matplotlib e Seaborn. La parte grafica è stata comunque ampiamente analizzata e migliorata con Power Bi.

Per non rendere la relazione troppo lunga ne riporto solo alcuni. Allego ovviamente tutto il codice per completezza informativa.

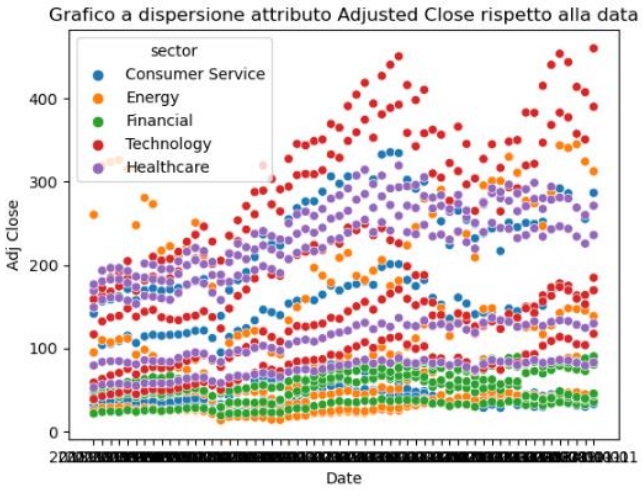
- Istogramma

```
In [14]:
plt.figure(figsize=(10, 6))
sns.histplot(df['Adj Close'], bins=20, kde=True)
plt.title('Adjusted Close Distribution')
plt.show()
```



- Grafico a Dispersione

```
In [13]: # Esempio di grafico a dispersione


sns.scatterplot(data=df, x='Date', y='Adj Close', hue='sector')
plt.title('Grafico a dispersione attributo Adjusted Close rispetto alla data')
plt.show()
```



2.2KPI

Al fine di avere degli indicatori rilevanti per analizzare in modo efficace e efficiente l'argomento trattato, sono state individuate e calculate diverse KPI rilevanti – sia con Python che con Excel.

Rilevanti KPI individuate e calcolate:

- Rendimento Cumulativo Percentuale
- Volatilità
- Rendimento Giornaliero Medio
- Max Drawdown
- Rendimento Annuo
- % cambiamento Prezzo Apertura e Chiusura

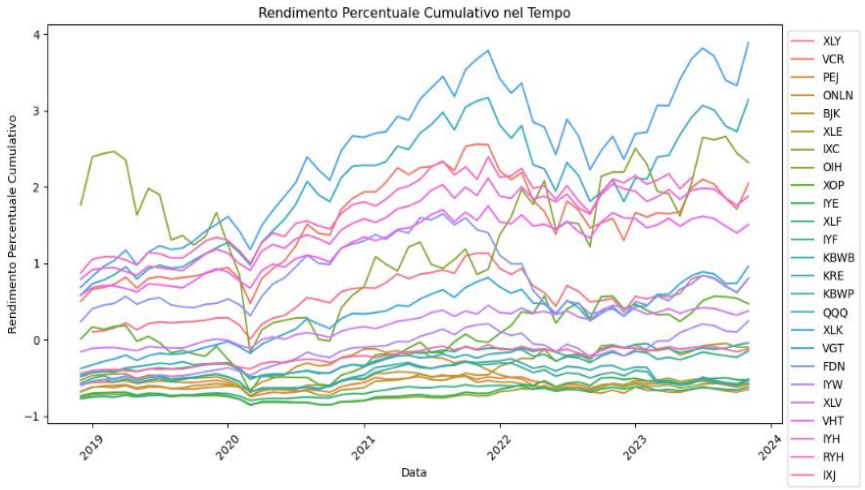
Riporto alcuni esempi di calcolo con Python:

- Rendimento Percentuale Cumulativo nel Tempo

Questo KPI rappresenta la crescita totale % di un investimento nell'orizzonte temporale selezionato (inclusi potenziali guadagni e perdite) – in altre parole, mostra la performance globale nel corso del tempo.

RENDIMENTO CUMULATIVO PERCENTUALE

```
In [20]: df['Cumulative Return'] = (1 + df['Adj Close'].pct_change()).cumprod() - 1

In [21]: plt.figure(figsize=(12, 6))
         sns.lineplot(data=df, x='Date', y='Cumulative Return', hue='ETF Name')
         plt.title('Rendimento Percentuale Cumulativo nel Tempo')
         plt.xlabel('Data')
         plt.ylabel('Rendimento Percentuale Cumulativo')
         plt.xticks(rotation=45)
         plt.legend(loc='upper left', bbox_to_anchor=(1, 1))
         plt.show()
```



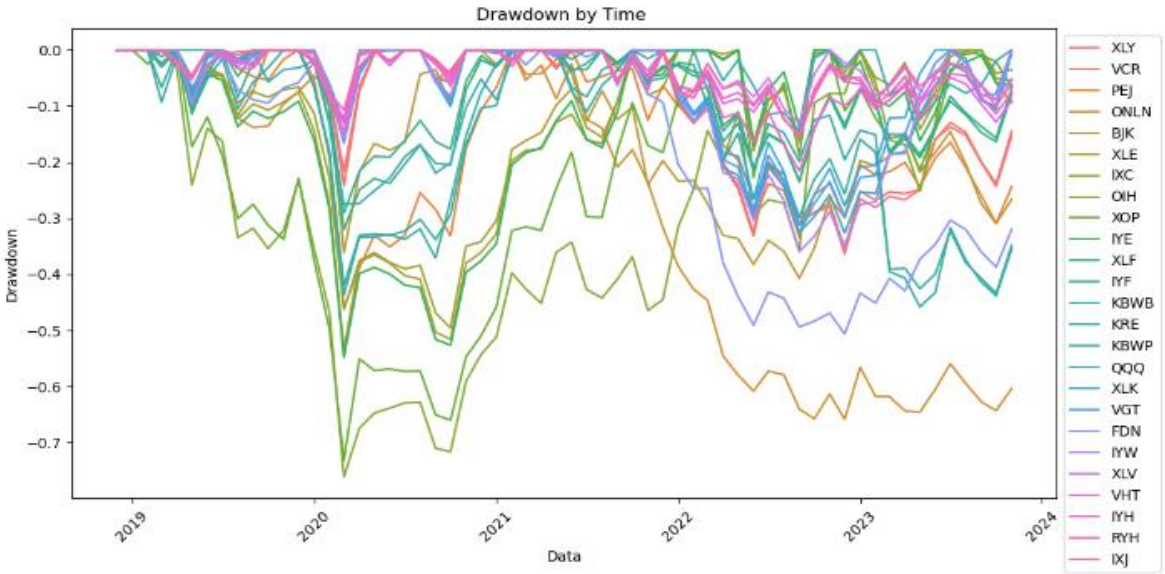
- Max Drawdown

Questo indicatore è stato scelto perché rappresenta la massima perdita % registrata dall'ETF – è un indicatore di rischio che mostra la perdita potenziale massima sperimentata dagli investitori.

MAX DRAWDOWN

```
[26]: df['Roll_Max'] = df['Adj Close'].cummax()
      df['Drawdown'] = df['Adj Close'] / df['Roll_Max'] - 1
      max_drawdown = df.groupby('ETF Name')['Drawdown'].min()
```

```
[27]: # Rappresentazione Grafica

      df['Date'] = pd.to_datetime(df['Date']) # mi assicura che la mia colonna Date sia in formato date usanc

      # Calcola il drawdown per ciascun ETF
      df['Roll_Max'] = df.groupby('ETF Name')['Adj Close'].cummax()
      df['Drawdown'] = df['Adj Close'] / df['Roll_Max'] - 1

      # Line-Plot nel tempo

      plt.figure(figsize=(12, 6))
      sns.lineplot(data=df, x='Date', y='Drawdown', hue='ETF Name') # due linee creano il grafico

      # Personalizzazione del grafico

      plt.title('Drawdown by Time ')
      plt.xlabel('Data')
      plt.ylabel('Drawdown')
      plt.xticks(rotation=45) # ruota le etichette di 45 gradi per rendere il tutto + leggibile
      plt.legend(loc='upper left', bbox_to_anchor=(1, 1))
      plt.show()
```



Per Quanto riguarda Excel e Power BI ho eseguito lo stesso tipo di calcoli utilizzano formule per Excel e Formule in M in PowerQuery.

3. DATA VISUALIZATION

Una volta scaricato il file definitivo .csv con anche le KPI calcolate - e prima di procedere alla realizzazione della Dashboard finale con Power BI, ho analizzato di nuovo il Dataset sia con Excel che con Power Query per verificare di nuovo qualora ci fosse stato data missing etc.

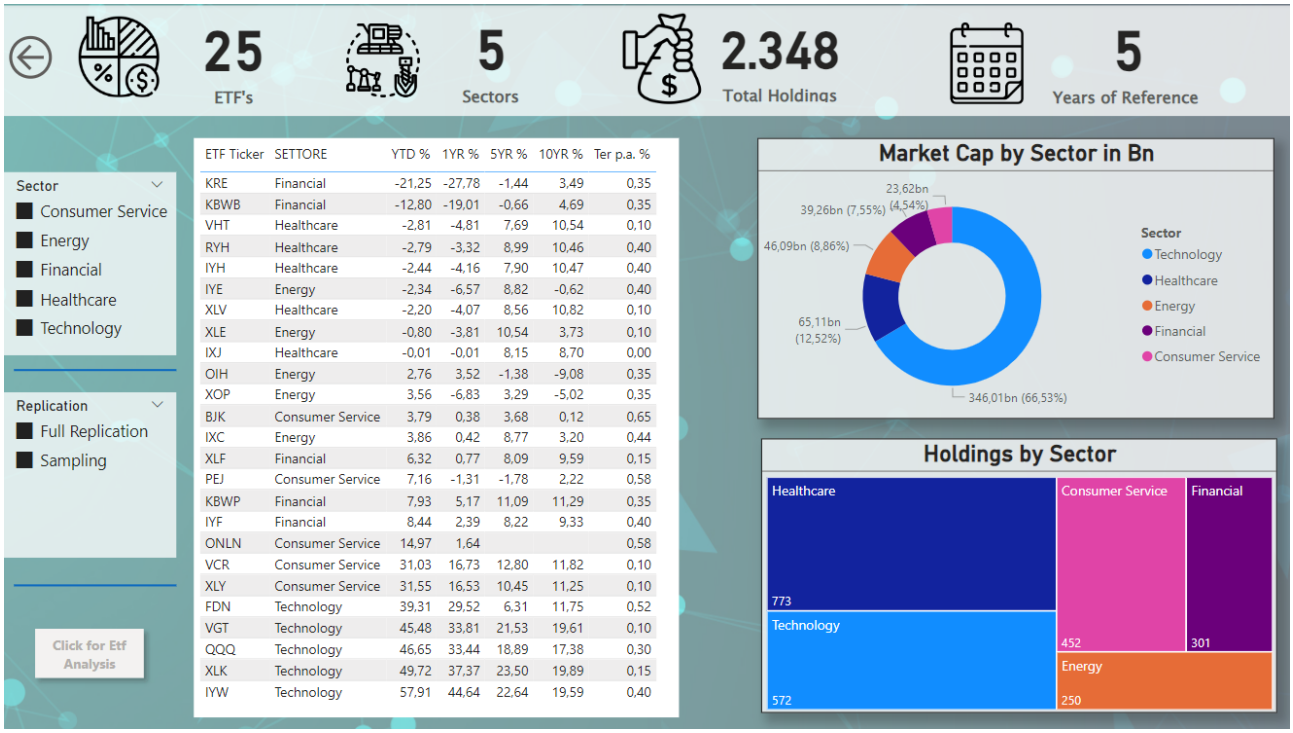
Le operazioni di ulteriore controllo e pulizia come per esempio verificare i data Type per ogni colonna sono stati svolti con Power Query per essere più rapido ed efficace.

Pulito e sistemato il Dataset ho iniziato la realizzazione delle visualizzazioni finali.

3.1 POWER BI

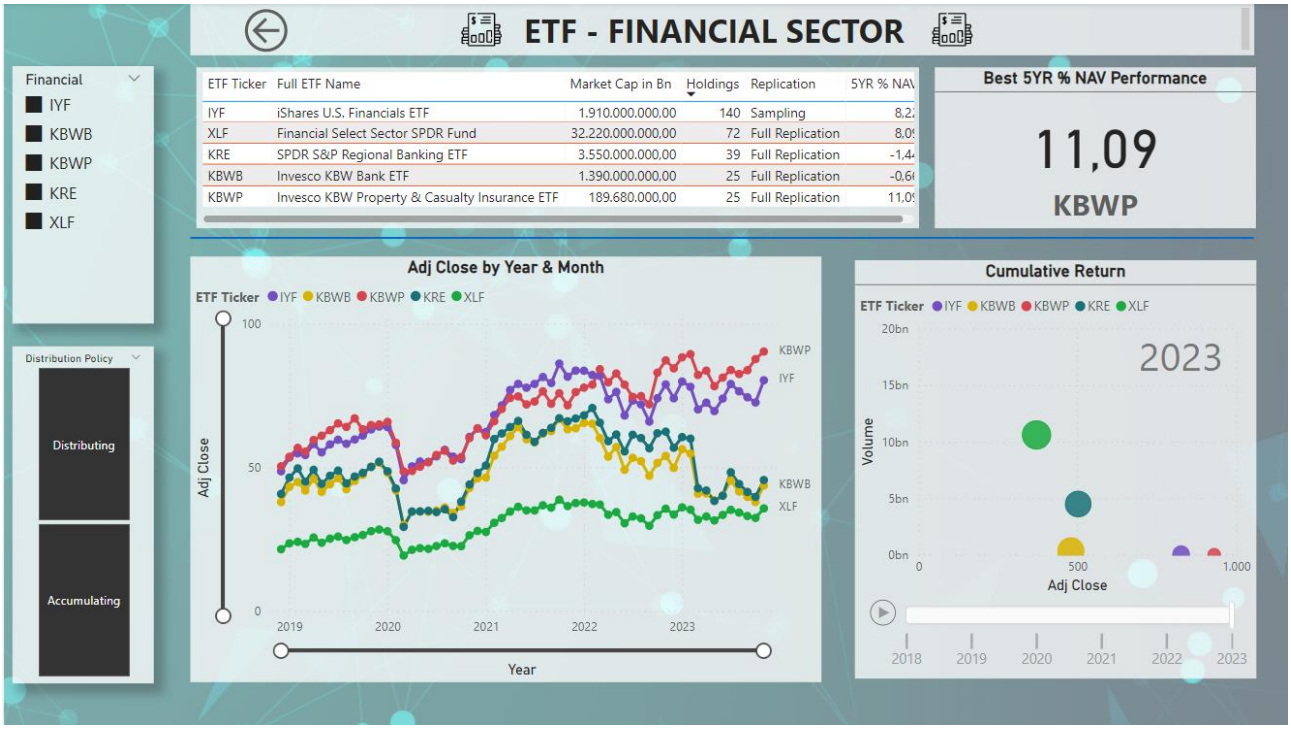
Per concludere il mio progetto di analisi ho deciso di utilizzare, come strumento di Data Visualization Power BI, che è stato una piacevolissima sorpresa per me nel corso di studio.

La dashboard finale è stata pensata in questo modo, una prima pagina di Overview dove ho inserito una serie di grafici e indicatori riassuntivi per spiegare a grandi linee gli ETF analizzati.



Le pagine seguenti sono una rappresentazione più approfondita per settore.

Esempio settore Finanziario:



- **Grafico Filtro**: Utilizzati due uno con tutti i ticker degli ETF del settore preso a riferimento e un altro con la Distribution Policy: Accumulating – Distributing.

- **Grafico Tabella**: Utilizzata per riassumere alcuni elementi per me importanti da rappresentare a questo livello di rappresentazione grafica, come ETF ticker, full name, Market Cap, Replication e NAV performance in 5 Anni.

- **Grafico Line Chart**: Perfetto per spiegare una visualizzazione che prevede un trend nel tempo – nel mio caso ho scelto il Pricing Adjusted Close per studiare l'andamento dei 5 etf settoriali per l'orizzonte temporale selezionato di 5 anni.

- **Grafico Card**: Con in rilievo il Ticker e il valore in % dell'ETF che ha performato meglio per KPI NAV % in 5 anni.

- **Grafico Scatter Chart Dinamico**: Prendo a riferimento il pricing Adjusted Close nell'asse x, il volume di scambio nell'asse Y, la grandezza della bolla è proporzionata per il Cumulative Return. Per renderlo dinamico, infine, ho inserito gli anni in Data nella Plax Axis.

Questo ripetuto con una scheda specifica per ogni settore selezionato ai fini dell'analisi.

Sono state inserite 3 schede supplementari:

1. ETF FOCUS: Utilizzando un Drill through dalla pagina Overview selezionando un ETF cliccando sul pulsante “Click for Etf Analysis” si apre dinamicamente la scheda in questione.

   Qui ho inserito:

- **Tabella riassuntiva** specifica per ETF selezionata.
- **Grafici Cards** con specifiche come replication, Inception Date, Distribution Policy Avg Volume.
- **Line chart** con andamento del closing price con andamento annuale e mensile.

2. TECHNICAL DATA SHEET: Inserita una scheda molto utile per avere info supplementari riguardo gli ETF selezionati come Benchmark e connessione a etf.com esterno tramite collegamento.

   Ho creato un nuovo campo direttamente su Power BI denominato Link Esterno – dove ho inserito una concatenazione tra ticker e collegamento a link esterno (qui la formula utilizzata → Link Esterno = "https://www.etf.com/"& KPI_etf_concatenati [ETF Ticker]).

3. NOTA METODOLOGICA: L’ultima scheda del progetto è una breve Nota dove spiego perché ho preso a riferimento alcune KPI invece di altre nella scelta delle metriche per la creazione dei Visuals.

CONCLUSIONI

In conclusione, il progetto di analisi degli ETF ha fornito una panoramica dettagliata sulle performance di vari ETF settoriali. Attraverso un processo completo di scraping dei dati, manipolazione e pulizia tramite Python, Excel e Power Query, e visualizzazione dei dati tramite Power BI, sono stati ottenuti risultati significativi.

L'analisi mi ha permesso di identificare tendenze e pattern nei dati, facilitando la comprensione delle dinamiche del mercato degli ETF in settori specifici come Tecnologia, Energia, Finanza, Sanità e Beni di Consumo. Le KPI selezionate, come, ad esempio, il rendimento cumulativo percentuale nel tempo, la volatilità e il max drawdown, hanno contribuito a valutare le performance degli ETF in modo efficace.

L'utilizzo di strumenti diversificati ha arricchito il processo analitico, consentendo una visione completa del panorama finanziario degli ETF. La dashboard finale in Power BI offre un'esperienza visiva intuitiva e interattiva, agevolando la comprensione dei risultati ottenuti.

È importante sottolineare che, nonostante l'approccio accurato e sistematico utilizzato, ogni analisi finanziaria ha i suoi limiti e dipende dalla qualità dei dati disponibili. Ulteriori sviluppi potrebbero includere l'implementazione di modelli predittivi o analisi più approfondite su specifici ETF di interesse.

In conclusione, questo progetto ha rappresentato un'opportunità per me di applicare in modo pratico e concreto le competenze acquisite durante il corso di Data Analyst presso Epicode School, offrendomi una visione approfondita delle dinamiche del mercato degli ETF e delle potenzialità degli strumenti analitici impiegati.