DataNest

May 24, 2025

Link al website: DataNest: the smart place for smart data (https://datanest1.streamlit.app/)

[ ]:
```
import streamlit as st
import pandas as pd
import matplotlib.pyplot
import plotly.express as px
import altair as alt
import numpy as np
from scipy.stats import zscore
import streamlit.components.v1 as components

# Inietta gli script di Microsoft Clarity e Google Tag per l'analisi del␣
 ↪comportamento degli utenti
st.markdown("""
<!-- Microsoft Clarity -->
<script type="text/javascript">
    (function(c,l,a,r,i,t,y){
        c[a]=c[a]||function(){(c[a].q=c[a].q||[]).push(arguments)};
        t=l.createElement(r);t.async=1;t.src="https://www.clarity.ms/tag/" + i;
        y=l.getElementsByTagName(r)[0];y.parentNode.insertBefore(t,y);
    })(window, document, "clarity", "script", "rnynbnc8ad");
</script>

<!-- Google tag (gtag.js) -->
<script async src="https://www.googletagmanager.com/gtag/js?id=G-SK988X9GTZ"></
 ↪script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag('js', new Date());
  gtag('config', 'G-SK988X9GTZ');
</script>
""", unsafe_allow_html=True)

# Titolo del sito web
st.title("DataNest: the smart place for smart data")

# Caricamento di un file CSV o XLSX nel sito web
```

```
file_caricato=st.file_uploader("Upload a CSV/XLSX file:", type=["csv", "xlsx"])

# Inizializzazione della session_state
if "dataset_originale" not in st.session_state:
    st.session_state.dataset_originale=None
if "dataset_modificato" not in st.session_state:
    st.session_state.dataset_modificato=None
# Inizializzazione della variabile per il popup di annullamento
if "mostra_popup_annulla" not in st.session_state:
    st.session_state.mostra_popup_annulla=False

if file_caricato is not None:
    if file_caricato.name.endswith(".csv"):
        st.session_state.dataset_originale=pd.read_csv(file_caricato)
    elif file_caricato.name.endswith(".xlsx"):
        st.session_state.dataset_originale=pd.read_excel(file_caricato)
    # Se il dataset originale è stato caricato, copia in dataset_modificato
    if st.session_state.dataset_originale is not None:
        st.session_state.dataset_modificato=st.session_state.dataset_originale.
 ↪copy()
    st.write("The file was uploaded successfully!")

    # Paragrafo n°1: Data Preview
    st.subheader("Data Preview")
    st.text("In this section, you can view and, if you wish, edit any data you␣
 ↪want.")
    scelta_dataset_modificabile=st.checkbox("Check here if you want to be able␣
 ↪to edit the attached file.")
    if scelta_dataset_modificabile==True:
        st.warning("  The file can be edited! Uncheck the option above if you␣
 ↪don't want that.")
        st.session_state.dataset_modificato=st.data_editor(st.session_state.
 ↪dataset_modificato, num_rows="dynamic")
    else:
        st.warning("  The file is NOT editable! Check the option above if you␣
 ↪want to change that.")
        st.dataframe(st.session_state.dataset_originale,␣
 ↪use_container_width=True)
    if scelta_dataset_modificabile==True:
        pulsante_annulla_modifiche=st.button("Cancel the changes on the file",␣
 ↪key="pulsante_annulla_modifiche")
        if pulsante_annulla_modifiche:
            st.session_state.mostra_popup_annulla=True
        if st.session_state.mostra_popup_annulla==True:
            with st.container():
```

```
                st.warning("  Are you sure you want to cancel all the changes␣
↪made to the file?")
                colonna_conferma_operazione, colonna_annulla_operazione=st.
↪columns(2)
                with colonna_conferma_operazione:
                    pulsante_colonna_conferma_operazione=st.button("OK, I'm␣
↪sure")
                    if pulsante_colonna_conferma_operazione==True:
                        st.session_state.dataset_modificato=st.session_state.
↪dataset_originale.copy()
                        st.success("  Changes successfully canceled!")
                        st.session_state.mostra_popup_annulla=False
                        st.write("The file has been restored to its original␣
↪version:")
                        st.dataframe(st.session_state.dataset_originale,␣
↪use_container_width=True)
                with colonna_annulla_operazione:
                        pulsante_colonna_annulla_operazione=st.button("No, stop␣
↪the current operation")
                        if pulsante_colonna_annulla_operazione==True:
                            st.success("  The operation has been canceled! The␣
↪changes to the file were not removed.")
                            st.session_state.mostra_popup_annulla=False

    # Paragrafo n°2: Data Summary
    st.subheader("Data Summary")
    st.write("In this section, you can view a summary of your data, including␣
↪basic statistics and information about the file.")
    if scelta_dataset_modificabile==True:
        dataset_finale=st.session_state.dataset_modificato
    else:
        dataset_finale=st.session_state.dataset_originale
    colonne_dataset_finale=dataset_finale.columns
    lista_colonne_dataset_finale=dataset_finale.columns.tolist()
    st.dataframe(dataset_finale.describe(), use_container_width=True)

    # Paragrafo n°3: Filter Data
    st.subheader("Filter Data")
    st.write("In this section, you can filter your data based on specific␣
↪values in one of the columns. Select a column and then a value to filter by.
↪")
    colonna_selezionata_menu=st.selectbox("Select a column to filter the data:
↪", lista_colonne_dataset_finale)
    valori_unici_dataset=dataset_finale[colonna_selezionata_menu].unique()
    valore_selezionato_menu=st.selectbox("Select a value to filter the data:",␣
↪valori_unici_dataset)
```

```
␣
↪dataset_finale_filtrato=dataset_finale[dataset_finale[colonna_selezionata_menu]==valore_sel
   st.dataframe(dataset_finale_filtrato, use_container_width=True)

   # Paragrafo n°4: NaN Data (Missing Values)
   st.subheader("NaN Data (Missing Values)")
   st.write("In this section, you can identify rows that contain missing␣
↪values (NaN) in any of the columns. This helps you quickly spot incomplete␣
↪or inconsistent data.")
   dataset_valori_NaN=dataset_finale[dataset_finale.isna().any(axis=1)]
   numero_NaN_presenti=len(dataset_valori_NaN)
   if dataset_valori_NaN.empty==True:
       st.success("  No missing values (NaN) found in the attached file!")
   else:
       if numero_NaN_presenti==1:
           st.error(f"  There is {numero_NaN_presenti} row that contains NaN␣
↪values (missing values):")
       else:
           st.error(f"  There are {numero_NaN_presenti} rows that contain NaN␣
↪values (missing values):")
       st.info('  The NaN value (missing value) in each row is shown as "None"␣
↪in the corresponding cell.')
       st.dataframe(dataset_valori_NaN, use_container_width=True)

   # Paragrafo n°5: Outliers Data
   st.subheader("Outliers Data")
   st.write("This section analyzes all numeric columns and shows the outliers␣
↪using Z-score method.")
   valore_soglia_outlier=st.slider("Set the Z-score threshold to detect␣
↪outliers:", 1.0, 5.0, 3.0)
   colonne_numeriche_dataset=dataset_finale.select_dtypes(include=np.number).
↪columns
   outliers_dataset=pd.DataFrame()
   for colonna_dataset in colonne_numeriche_dataset:
       serie=dataset_finale[colonna_dataset]
       z_scores=pd.Series(zscore(serie.dropna()), index=serie.dropna().index)
       mask=np.abs(z_scores)>valore_soglia_outlier
       outliers_colonna_dataset = dataset_finale.
↪loc[dataset_finale[colonna_dataset].dropna().index[mask]].copy()
       if not outliers_colonna_dataset.empty==True:
           outliers_colonna_dataset["Outlier Column"]=colonna_dataset
           outliers_dataset=pd.concat([outliers_dataset,␣
↪outliers_colonna_dataset])
   numero_righe_outliers=len(outliers_dataset)
   if not outliers_dataset.empty==True:
       if numero_righe_outliers==1:
```

```
            st.error(f"  Found {numero_righe_outliers} outlier row in total in␣
↪the uploaded file!")
        else:
            st.error(f"  Found {numero_righe_outliers} outlier rows in total in␣
↪the uploaded file!")
        st.info('  The name of the column containing the outlier value for each␣
↪row is shown in the "Outlier Column" column.')
        st.dataframe(outliers_dataset, use_container_width=True)
    else:
        st.success("  No outliers found in this file based on the selected␣
↪Z-score threshold!")

    # Paragrafo n°6: Data Visualization
    st.subheader("Data Visualization")
    st.write("In this section, you can create different types of charts to␣
↪visualize the data. Choose the columns you want to plot and select the chart␣
↪type.")
    valore_asse_x_colonna=st.selectbox("Select the column to use for the X axis␣
↪in the chart:", lista_colonne_dataset_finale)
    valore_asse_y_colonna=st.selectbox("Select the column to use for the Y axis␣
↪in the chart:", lista_colonne_dataset_finale)
    pulsante_genera_grafici=st.button("Generate plots (charts)")
    if pulsante_genera_grafici==True:
        if valore_asse_x_colonna==valore_asse_y_colonna:
            st.error("  Please select different columns for the X and Y axes.␣
↪You have selected the same column for both axes.")
        elif valore_asse_x_colonna not in colonne_dataset_finale or␣
↪valore_asse_y_colonna not in colonne_dataset_finale:
            st.error("  One or both of the selected columns do not exist in the␣
↪file. Please try again with valid column names.")
        else:
            st.write("The following charts have been generated based on your␣
↪selection:")
            # Grafici a linee
            st.write("1) Line charts:")
            st.line_chart(dataset_finale.
↪set_index(valore_asse_x_colonna)[valore_asse_y_colonna])
            grafico_linea_interattivo=px.line(dataset_finale,␣
↪x=valore_asse_x_colonna, y=valore_asse_y_colonna, markers=True)
            st.plotly_chart(grafico_linea_interattivo)
            # Grafici a barre
            st.write("2) Bar charts:")
            st.bar_chart(dataset_finale.
↪set_index(valore_asse_x_colonna)[valore_asse_y_colonna])
```

```
            grafico_barre_ordinato=px.bar(dataset_finale.
 ↪sort_values(by=valore_asse_y_colonna), x=valore_asse_y_colonna,␣
 ↪y=valore_asse_x_colonna, orientation="h")
            st.plotly_chart(grafico_barre_ordinato)
            # Grafici a aree
            st.write("3) Area charts:")
            st.area_chart(dataset_finale.
 ↪set_index(valore_asse_x_colonna)[valore_asse_y_colonna])
else:
    st.write("Waiting for file upload...")
    st.write("No file uploaded yet.")
```