Northeastern University

INFO 6105

Data Science Engineering Method

Capstone Project

Streamlit Project and AI generated song album

Name - Dharun Karthick Ramaraj
NUID – 002878156

Table of Contents

1) **Abstract**
    a. Overview of the project and its purpose.
2) **Introduction**
    a. Background information and significance of the project.
3) **Technologies Used**
    a. Detailed list and explanation of the technologies and libraries employed in the project.
4) **Installation Command**
    a. Step-by-step instructions for setting up the project environment.
5) **Additional Setup Notes**
    a. Recommendations for Python version, H2O specifics, and virtual environment setup.
6) **Core Functionalities**
    a. Detailed overview of the key features and capabilities of the app.
    b. Subsections:
        i. Data Upload and Preparation
        ii. Exploratory Data Analysis (EDA)
        iii. Model Building and Evaluation
7) **Usage Guide**
    a. Comprehensive guide on how to use the app effectively.
    b. Subsections:
        i. Launching the App
        ii. Uploading Data
        iii. Preparing Your Data
        iv. Exploratory Data Analysis (EDA)
        v. Building and Evaluating Models
        vi. Advanced Features
        vii. Closing the Session

8) **Screenshot**
    a. Visual aids to illustrate key aspects of the app interface and functionalities.
9) **Code**
    a. Full source code of the application, with detailed explanations of major functions and their roles within the app.
10) **Code Explanation**
    a. In-depth explanations of the source code, describing the purpose and functionality of main functions and classes.
    b. Subsections:
        i. Main Function
        ii. Perform EDA Function
        iii. Run Models Function
        iv. Run H2O AutoML Function
11) **Future Improvements**
    a. A look ahead at possible enhancements that could be made to the project to increase functionality and user engagement.
12) **Conclusion**
    a. Summary of the project's contributions to the field of data science and potential impact on future data-driven decision making.

Abstract

The "Data Exploration and Machine Learning Modeling" app is a comprehensive tool designed for data scientists and analysts to streamline the process of data analysis and predictive modeling. Built with Streamlit, this app allows users to upload CSV files, conduct detailed exploratory data analysis (EDA), and utilize a variety of machine learning algorithms to develop robust models. Key functionalities include the ability to dynamically drop data columns, perform visual EDA on both numeric and categorical data, and choose from a selection of traditional machine learning models or leverage H2O's AutoML capabilities for automated model selection and training. The app also integrates SHAP values for interpretability of model predictions, making it an invaluable tool for both exploratory and predictive analysis. Designed to facilitate a deeper understanding of data features and improve prediction accuracy, this app serves as an essential resource for anyone looking to enhance their data-driven decision-making processes.

Introduction

In the ever-evolving field of data science, the ability to rapidly analyze data and generate predictive models is crucial for making informed decisions. The " Data Exploration and Machine Learning Modeling" app, developed using Streamlit, is designed to address these needs by providing a user-friendly interface for data exploration and automated machine learning.

Technologies Used

1. Streamlit: For creating and running the web app interface.
2. Pandas: For data manipulation and analysis.
3. NumPy: For numerical operations.
4. Seaborn: For statistical data visualization.
5. Matplotlib: For creating additional plots.
6. H2O: For running AutoML for machine learning tasks.
7. SHAP: For calculating SHapley Additive exPlanations to interpret model outputs.
8. Scikit-learn: For traditional machine learning models and metrics.

Installation Command

- pip install streamlit pandas numpy seaborn matplotlib h2o shap scikit-learn

Additional Setup Notes

Python Version: Ensure that you are using a Python version compatible with all these libraries, typically Python 3.7 or later.

H2O Specifics: H2O runs a Java backend, so ensure that you have Java installed on your machine. You can check this by running java -version in your command prompt or terminal.

Virtual Environment: It's a good practice to use a virtual environment for Python projects to manage dependencies without conflicts

Core Functionalities

1. **Data Upload and Preparation**: Users can upload their CSV files directly into the app. The app provides options to view the raw data and select columns to be dropped, enabling preprocessing of data to fit the user's needs.
2. **Exploratory Data Analysis (EDA)**: The app offers tools to conduct comprehensive EDA. Users can select numeric or categorical columns and visualize distributions and counts, respectively. This aids in uncovering underlying patterns or anomalies in the data.
3. **Model Building and Evaluation**: Users can choose between traditional machine learning models and H2O's AutoML for model building. The app supports both regression and classification problems and provides metrics to evaluate model performance. For models capable of it, SHAP values are calculated and displayed to interpret the effects of the variables.

Usage Guide

Step 1: Launching the App

- **Accessing the App**: Open your web browser and navigate to the URL where the app is hosted. This could be a local address (typically **localhost**) if you're running it on your own machine, or a public URL if it's deployed on a server.

Step 2: Uploading Data

- **File Upload**: Click on the **Choose a CSV file** uploader to select and upload your dataset. Ensure that the dataset is in CSV format as this is what the app is configured to process.
- **Data Preview**: Once uploaded, you can view the first few rows of your dataset by checking the 'Show raw data' box, which will display a preview of the data.

Step 3: Preparing Your Data

- **Select Columns to Drop**: If there are columns that you wish to exclude from analysis, check the 'Select Columns to Drop' box. A dropdown will appear allowing you to select one or more columns that you want to remove from the dataset. The updated dataset will be displayed under 'Updated Data Preview'.

Step 4: Exploratory Data Analysis (EDA)

- **Initiating EDA**: Check the 'Show EDA' box to start the exploratory data analysis process.

- **Numeric EDA**: Select the numeric columns for which you want to see distributions by choosing from a multiselect dropdown list. Histograms will be generated for each selected column.
- **Categorical EDA**: Similarly, select categorical columns to view the count of categories. Bar charts will be displayed for each selected column.

Step 5: Building and Evaluating Models

- **Select Problem Type**: Choose between 'Regression' and 'Classification' depending on the nature of your target variable.
- **Select Target Variable**: Choose the variable you want to predict from another dropdown menu.
- **Model Selection and Training**:
    - If you opt to use traditional machine learning models, select one from the list provided (e.g., Linear Regression, SVM).
    - Alternatively, click on the 'Use AutoML' button to use H2O's AutoML feature. This will automatically select the best model based on your data.
- **View Results**: After model training, performance metrics such as MSE, RMSE, accuracy, or confusion matrix (depending on the problem type) will be displayed. If applicable, SHAP values can also be viewed to understand feature importances.

Step 6: Advanced Features

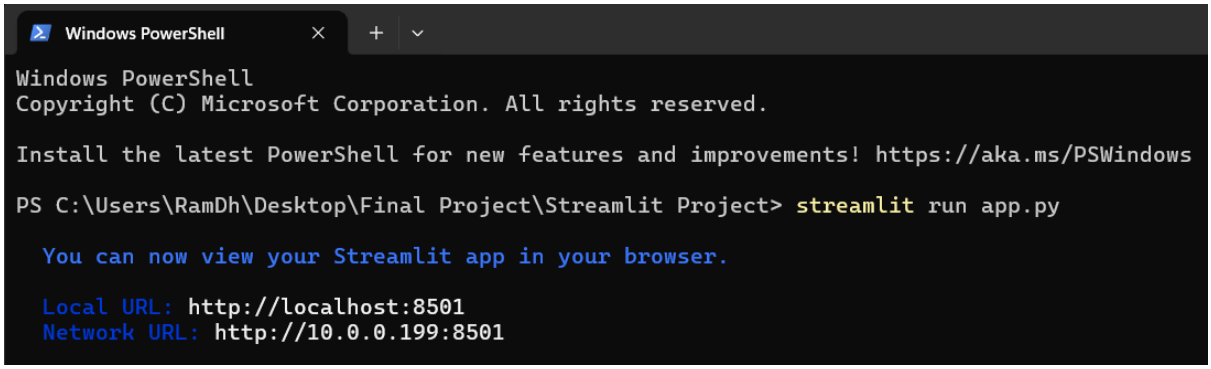
- **Interpreting Models with SHAP**: For compatible models, SHAP values will be calculated and displayed. This provides insights into how each feature contributes to the prediction, helping in understanding model behavior.

Step 7: Closing the Session

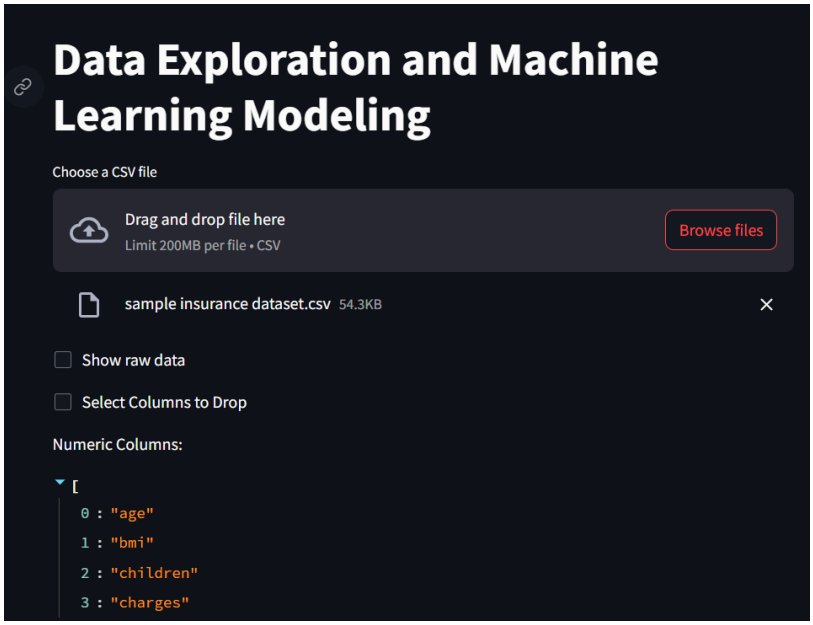
- **Save Your Work**: Ensure to save any outputs or models if the app supports this feature.
- **End Session**: Close your browser tab or window to end the session.

Screenshot

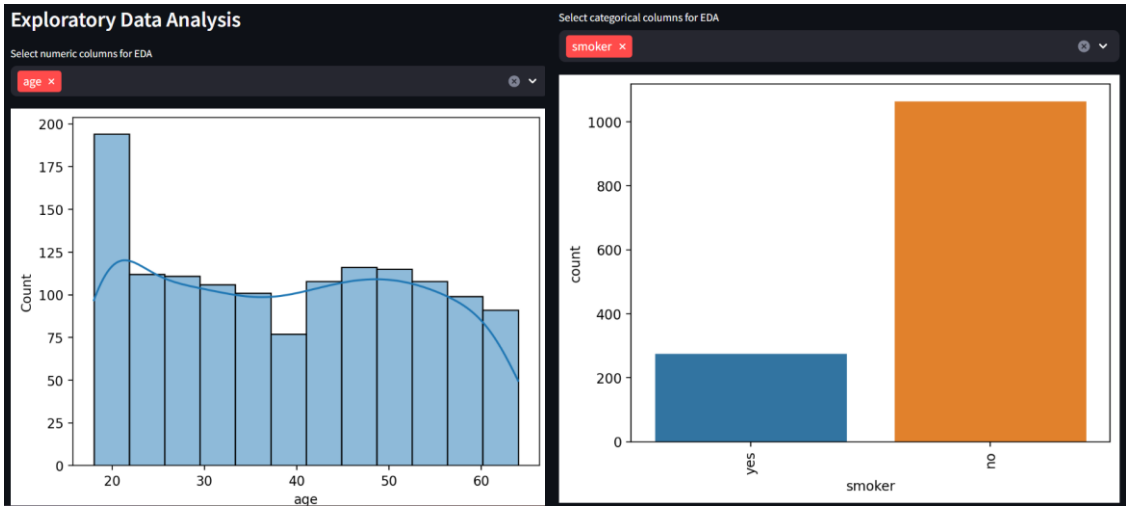
Running the app using Streamlit



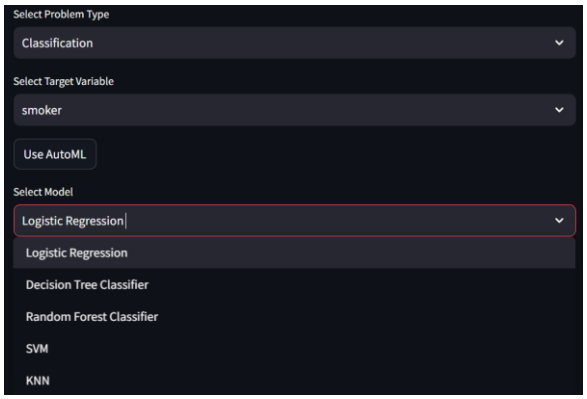
Uploading the dataset in the app

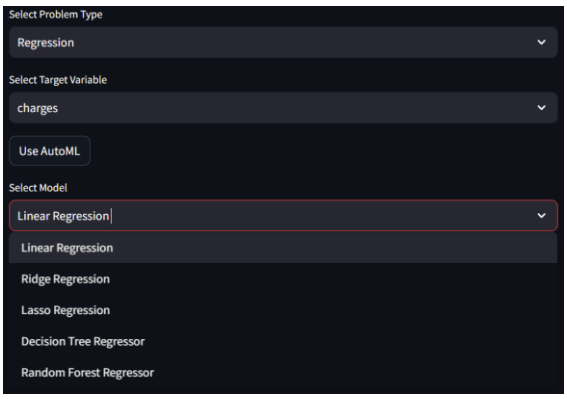


Testing the EDA option

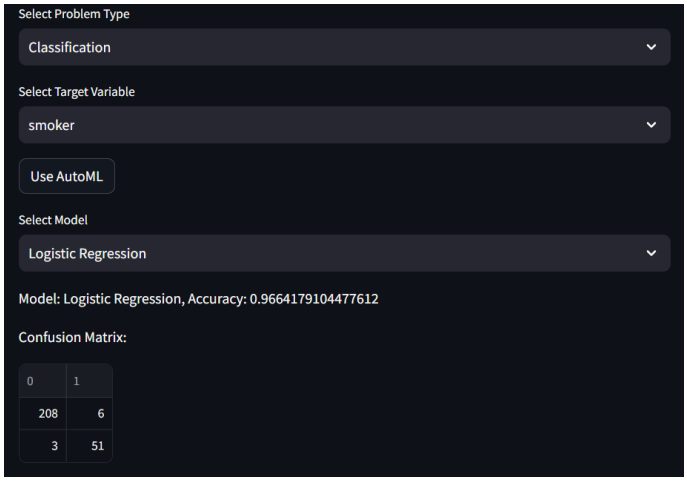


We can choose what model should be trained based on the problem from the below list

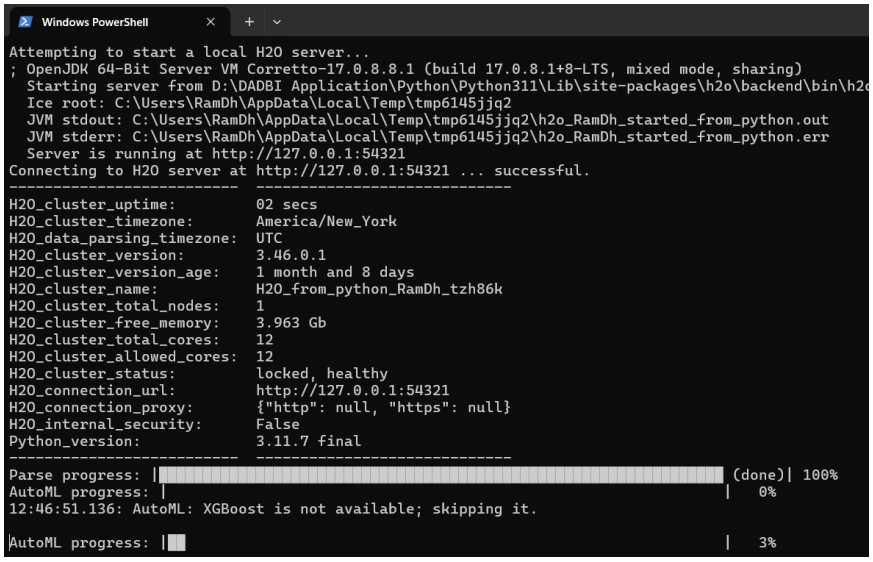


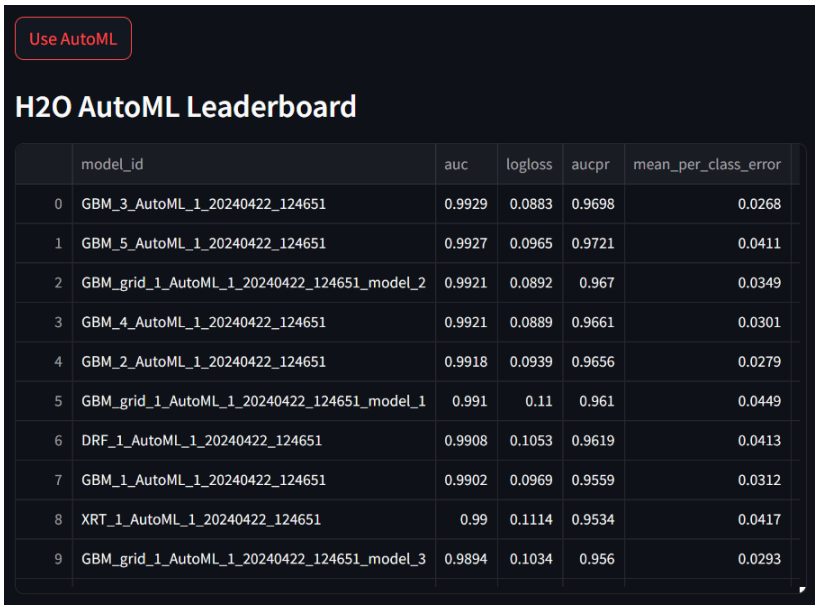


Using logistic regression on smoker feature



Using AutoML





	model_id	auc	logloss	aucpr	mean_per_class_error
0	GBM_3_AutoML_1_20240422_124651	0.9929	0.0883	0.9698	0.0268
1	GBM_5_AutoML_1_20240422_124651	0.9927	0.0965	0.9721	0.0411
2	GBM_grid_1_AutoML_1_20240422_124651_model_2	0.9921	0.0892	0.967	0.0349
3	GBM_4_AutoML_1_20240422_124651	0.9921	0.0889	0.9661	0.0301
4	GBM_2_AutoML_1_20240422_124651	0.9918	0.0939	0.9656	0.0279
5	GBM_grid_1_AutoML_1_20240422_124651_model_1	0.991	0.11	0.961	0.0449
6	DRF_1_AutoML_1_20240422_124651	0.9908	0.1053	0.9619	0.0413
7	GBM_1_AutoML_1_20240422_124651	0.9902	0.0969	0.9559	0.0312
8	XRT_1_AutoML_1_20240422_124651	0.99	0.1114	0.9534	0.0417
9	GBM_grid_1_AutoML_1_20240422_124651_model_3	0.9894	0.1034	0.956	0.0293

Code

```
import streamlit as st
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import h2o
from h2o.automl import H2OAutoML
import shap
import math

from sklearn.model_selection import train_test_split
```

```
from sklearn.linear_model import LinearRegression, LogisticRegression, Ridge,
Lasso
from sklearn.tree import DecisionTreeRegressor, DecisionTreeClassifier
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import mean_squared_error, mean_absolute_error,
accuracy_score, confusion_matrix

def main():
    st.title('Data Exploration and Machine Learning Modeling')

    uploaded_file = st.file_uploader("Choose a CSV file", type="csv")
    if uploaded_file is not None:
        data = pd.read_csv(uploaded_file)
        if st.checkbox('Show raw data'):
            st.write("Data Preview:", data)

        if st.checkbox('Select Columns to Drop', False):
            columns_to_drop = st.multiselect('Select columns to drop',
data.columns)
            data = data.drop(columns=columns_to_drop)
            st.write("Updated Data Preview:", data)

        numeric_columns =
data.select_dtypes(include=np.number).columns.tolist()
        categorical_columns =
data.select_dtypes(exclude=np.number).columns.tolist()
        st.write("Numeric Columns:", numeric_columns)
        st.write("Categorical Columns:", categorical_columns)

        if st.checkbox('Show EDA', False):
            perform_eda(data, numeric_columns, categorical_columns)

        if st.checkbox('Build Model'):
            problem_type = st.selectbox('Select Problem Type', ['Regression',
'Classification'])
            target_variable = st.selectbox('Select Target Variable',
numeric_columns if problem_type == 'Regression' else categorical_columns)
            if target_variable:
                if st.button("Use AutoML"):
                    h2o.init()  # Initialize H2O when the AutoML button is
clicked
                    run_h2o_automl(data, target_variable, problem_type)
                else:
                    run_models(data, numeric_columns, categorical_columns,
target_variable, problem_type)
```

```
def perform_eda(data, numeric_columns, categorical_columns):
    st.subheader('Exploratory Data Analysis')
    selected_numeric_columns = st.multiselect('Select numeric columns for
EDA', numeric_columns)
    if selected_numeric_columns:
        for column in selected_numeric_columns:
            fig, ax = plt.subplots()
            sns.histplot(data[column], kde=True, ax=ax)
            st.pyplot(fig)

    selected_categorical_columns = st.multiselect('Select categorical columns
for EDA', categorical_columns)
    if selected_categorical_columns:
        for column in selected_categorical_columns:
            fig, ax = plt.subplots()
            sns.countplot(x=data[column], ax=ax)
            plt.xticks(rotation=90)
            st.pyplot(fig)

def run_models(data, numeric_columns, categorical_columns, target_variable,
problem_type):
    X = data[numeric_columns].drop(columns=[target_variable]).fillna(0) if
problem_type == 'Regression' else pd.get_dummies(data[categorical_columns +
numeric_columns].drop(columns=[target_variable]), drop_first=True)
    y = data[target_variable]

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

    models = {
    'Regression': [
        ('Linear Regression', LinearRegression()),
        ('Ridge Regression', Ridge()),
        ('Lasso Regression', Lasso()),
        ('Decision Tree Regressor', DecisionTreeRegressor()),
        ('Random Forest Regressor', RandomForestRegressor())
    ],
    'Classification': [
        ('Logistic Regression', LogisticRegression(max_iter=1000)),
        ('Decision Tree Classifier', DecisionTreeClassifier()),
        ('Random Forest Classifier', RandomForestClassifier()),
        ('SVM', SVC()),
        ('KNN', KNeighborsClassifier())
    ]
}


    selected_model = st.selectbox('Select Model', [model[0] for model in
models[problem_type]])
```

```
    model = dict(models[problem_type])[selected_model]
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)

    if problem_type == 'Regression':
        mse = mean_squared_error(y_test, predictions)
        rmse = math.sqrt(mse)
        mae = mean_absolute_error(y_test, predictions)
        st.write(f'Model: {selected_model}, MSE: {mse}, RMSE: {rmse}, MAE:
{mae}')
    else:
        accuracy = accuracy_score(y_test, predictions)
        cm = confusion_matrix(y_test, predictions)
        st.write(f'Model: {selected_model}, Accuracy: {accuracy}')
        st.write('Confusion Matrix:', cm)

def run_h2o_automl(data, target_variable, problem_type):
    h2o_df = h2o.H2OFrame(data)
    x = [name for name in h2o_df.columns if name != target_variable]
    y = target_variable
    if problem_type == 'Classification':
        h2o_df[y] = h2o_df[y].asfactor()

    train, test = h2o_df.split_frame(ratios=[.8], seed=42)
    aml = H2OAutoML(max_models=15, max_runtime_secs=120, seed=1)
    aml.train(x=x, y=y, training_frame=train)

    st.subheader('H2O AutoML Leaderboard')
    st.write(aml.leaderboard.as_data_frame())
    st.subheader('Best Model Performance')
    st.write(aml.leader.model_performance(test))

    if isinstance(aml.leader, h2o.estimators.H2OTreeModel):
        explainer = shap.TreeExplainer(aml.leader)
        shap_values = explainer.shap_values(h2o.as_list(test))
        st.pyplot(shap.summary_plot(shap_values, h2o.as_list(test),
plot_type="bar"))

if __name__ == "__main__":
    main()
```

Code Explanation

Main Function: main()

- **Purpose**: Serves as the entry point for the Streamlit app. This function orchestrates the UI layout and calls other functions based on user interaction.
- **Key Components**:
    - **File Uploader**: Uses **st.file_uploader** to allow users to upload CSV files.
    - **Data Display**: Utilizes **st.checkbox** to toggle the visibility of the raw data uploaded by the user.
    - **Column Management**: Provides functionality to drop unnecessary columns through **st.multiselect**, which updates the DataFrame accordingly.
    - **Data Type Identification**: Identifies numeric and categorical columns in the DataFrame and displays them using **st.write**.
    - **Conditional Sections**: Based on user selections, this function may call **perform_eda** or **run_models** to perform further analysis or model building.

Function: perform_eda(data, numeric_columns, categorical_columns)

- **Purpose**: Handles the exploratory data analysis part of the app, creating visualizations for selected columns.
- **Key Components**:
    - **Numeric EDA**: For selected numeric columns, generates histograms using Seaborn's **sns.histplot** to show distributions.
    - **Categorical EDA**: For selected categorical columns, produces bar charts using Seaborn's **sns.countplot** to show the frequency of categories.
    - **Visualization Rendering**: Each plot is rendered within the Streamlit interface using **st.pyplot**.

Function: run_models(data, numeric_columns, categorical_columns, target_variable, problem_type)

- **Purpose**: Manages the machine learning model selection, training, and evaluation process.
- **Key Components**:
    - **Data Preparation**: Prepares feature matrix **X** and target vector **y** by excluding the target variable and handling missing values.

- **Model Selection**: Allows users to choose from a predefined list of machine learning models based on the problem type (regression or classification).
- **Model Training and Evaluation**: Trains the selected model and evaluates it using appropriate metrics (MSE, RMSE, MAE for regression; accuracy and confusion matrix for classification).
- **Results Display**: Outputs the performance metrics and confusion matrix in the Streamlit UI.

Function: run_h2o_automl(data, target_variable, problem_type)

- **Purpose**: Integrates H2O's AutoML capabilities for automated model selection and training.
- **Key Components**:
  - **H2O Initialization and Frame Conversion**: Initializes an H2O session and converts the pandas DataFrame to an H2O Frame.
  - **AutoML Configuration and Execution**: Configures and runs the AutoML process, specifying parameters like the maximum number of models and runtime.
  - **Leaderboard and Performance Display**: Displays the AutoML leaderboard and the best model's performance metrics.
  - **SHAP Value Calculation**: If the best model supports it, calculates and displays SHAP values to interpret the model's predictions.

Future Improvements

1. Expanded Model Library

- **Enhancement Description**: Incorporating a broader range of machine learning algorithms, including advanced models like XGBoost, LightGBM, and neural networks, would provide users with more powerful tools to handle complex data challenges. Additionally, allowing users to define and integrate their custom models would cater to advanced data scientists looking for more flexibility and control over their predictive modeling.
- **Benefits**: This would not only improve the app's versatility in solving a wider array of problems but also potentially increase the accuracy and efficiency of the models available to users.

2. Interactive Visualizations and Dashboard Customization

- **Enhancement Description**: Enhancing the app with more interactive visualization tools would allow users to dynamically interact with their data, such as adjusting parameters in real-time and seeing immediate results. Dashboard customization options would enable users to personalize their workspace, arranging tools and outputs in a way that best suits their workflow.
- **Benefits**: Interactive elements and customized layouts can significantly enhance user engagement, make the analysis process more intuitive, and allow for deeper insights by exploring data through various visual perspectives.

Conclusion

The "Advanced Data Exploration and Machine Learning Modeling" Streamlit app stands as a robust and user-friendly platform that significantly streamlines the process of data analysis and predictive modeling. Designed with a focus on accessibility and efficiency, this application empowers users, from data scientists to analysts, by providing advanced tools for uploading datasets, conducting exploratory data analysis, and building a variety of machine learning models. The integration of both traditional machine learning techniques and automated machine learning through H2O AutoML, coupled with interpretability features like SHAP values, ensures that users can not only develop but also understand their models in-depth.

Looking forward, the proposed enhancements aimed at expanding the model library, incorporating interactive visualizations, and improving performance through asynchronous processing will further elevate the app's functionality and user engagement. By continuing to evolve in response to user feedback and advances in technology, the app is poised to remain an essential tool in the data science community, facilitating insightful analyses and driving informed decision-making processes.

In conclusion, this Streamlit app embodies a comprehensive solution for data exploration and machine learning, making advanced analytics accessible and actionable for a broad spectrum of users. As the app progresses, it will undoubtedly continue to inspire and support innovative data-driven strategies across various sectors