Machine Learning Model Selection

Deployed at: https://mlmodelselection.streamlit.app/

Junaid Saleem
2022243
BS. Computer Science

***Abstract*—This project focuses on developing a user-friendly web application for machine learning tasks. The app integrates various preprocessing techniques and machine learning models, enabling users to interactively upload datasets, preprocess data, and evaluate both regression and classification models. The implementation utilizes the Streamlit framework for its simplicity and rapid prototyping capabilities, while integrating state-of-the-art Python libraries like scikit-learn, XGBoost, and CatBoost.**

***Index Terms*—Machine Learning, Data Preprocessing, Regression, Classification, Streamlit.**

I. Report Organization

The report begins with an **Introduction** that provides an overview of the project aimed at simplifying machine learning workflows. It highlights the **Problem Statement**, emphasizing the challenges faced by users, particularly those without technical expertise, in selecting and evaluating suitable machine learning models for their data. The introduction stresses the need for an intuitive system to bridge the gap between data preparation and model performance evaluation. The **Literature Review** covers regression and classification models, discussing their theoretical underpinnings and practical applications. The **Methods and Functionality** section describes the app's workflow, including dataset uploading, preprocessing options, feature visualization, model selection, and hyperparameter tuning for both regression and classification tasks. The **Results** section presents comparative metrics for various models, showcasing the app's ability to guide users in making informed decisions about model selection. Finally, the report concludes with a **Conclusion and Future Work** section, summarizing the project's contributions to accessible model selection and proposing further enhancements to expand its capabilities.

II. Introduction

A. Problem Statement

Selecting the most appropriate machine learning model is crucial for achieving optimal results, as different models are tailored to handle specific types of data more effectively. However, testing a dataset across multiple models to identify the best-performing one can be a time-consuming and resource-intensive process. This challenge is exacerbated for individuals with limited technical expertise in machine learning, making it difficult for them to navigate the complexities of model selection and evaluation. Therefore, there is a pressing need for a no-code solution that can guide users through the process of selecting the most suitable regression and classification models for their datasets. Such a system would streamline the model selection process, save valuable time, and empower users to make informed decisions without requiring in-depth knowledge of machine learning.

III. Literature Review

A. Data Preprocessing

Data preprocessing plays a vital role in preparing raw data for analysis and modeling. Common techniques include feature selection, normalization, data imputation (filling missing values), one-hot encoding for categorical variables, and feature visualization to uncover patterns and relationships within the data. These preprocessing steps ensure that the data is clean, normalized, and appropriately formatted for the models, improving their performance and accuracy.

B. Regression Models

Several regression models have been employed to predict continuous outcomes. Multiple Linear Regression is one of the simplest yet widely used techniques for predicting a dependent variable based on multiple independent variables. Polynomial Regression extends this concept by introducing higher-degree polynomial terms to capture nonlinear relationships. Support Vector Regression (SVR) is another powerful method that uses hyperplanes in higher-dimensional spaces to predict continuous variables. Decision Trees, known for their interpretability, and Random Forests, an ensemble of decision trees, are also commonly applied to regression tasks. More advanced models such as XGBoost and CatBoost, based on gradient boosting techniques, have gained popularity due to their efficiency, scalability, and high performance in handling complex datasets with non-linear relationships.

C. Classification Models

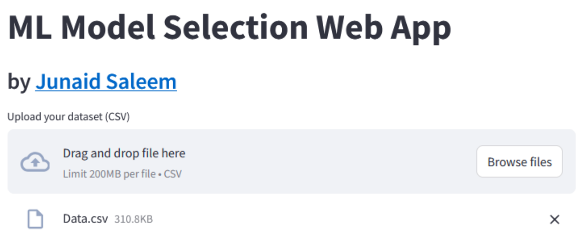
For classification tasks, a variety of models have been used to predict categorical outcomes. Logistic Regression is a statistical method used for binary classification problems, while Support Vector Machines (SVM) are known for their ability to handle high-dimensional data and find optimal hyperplanes for classification. Decision Trees and Random Forests have been extensively applied for both regression and classification, with Random Forests offering the advantage of reducing overfitting through ensemble learning. Naive Bayes, a probabilistic classifier based on Bayes' theorem, is particularly useful when the features are independent. k-Nearest Neighbors (KNN) is a simple yet effective model for classification based on proximity. Like regression tasks, XGBoost and CatBoost have been widely used in classification problems due to their robustness and performance in handling imbalanced datasets.

IV. Methods and Functionality

This section describes the methodology and functionality of the developed web application, outlining the process flow from uploading the dataset to obtaining results for both regression and classification tasks. The app is designed to guide users through the entire machine learning workflow, providing an intuitive interface for dataset manipulation, model selection, and performance evaluation.

A. Uploading the Dataset

The first step in the workflow is uploading the dataset. Users can easily upload their dataset in .csv format. The app supports seamless file handling and ensures the dataset is properly loaded for subsequent processing. This section allows users to directly interact with their data in the web interface, making it accessible for analysis.



ML Model Selection Web App

by Junaid Saleem

Upload your dataset (CSV)

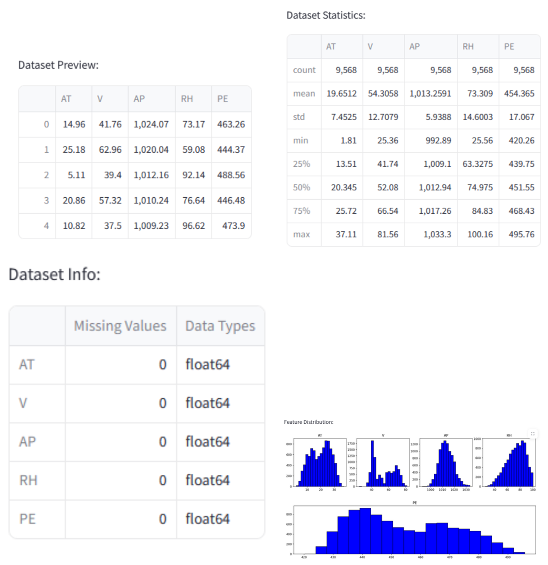
Drag and drop file here
Limit 200MB per file • CSV

Data.csv 310.8KB

Fig. 1. Dataset Upload Interface

B. Dataset Preview

Once the dataset is uploaded, users can preview the data. This includes displaying the first five rows of the dataset to give a quick overview of its structure. Additionally, the app shows statistical summaries such as the distribution of features, and minimum, maximum, mean and standard deviation values of each numerical feature. This preview helps users understand the dataset's contents and identify potential issues before proceeding with further preprocessing.



Dataset Preview:

	AT	V	AP	RH	PE
0	14.96	41.76	1,024.07	73.17	463.26
1	25.18	62.96	1,020.04	59.08	444.37
2	5.11	39.4	1,012.16	92.14	488.56
3	20.86	57.32	1,010.24	76.64	446.48
4	10.82	37.5	1,009.23	96.62	473.9

Dataset Statistics:

	AT	V	AP	RH	PE
count	9,568	9,568	9,568	9,568	9,568
mean	19.6512	54.3058	1,013.2591	73.309	454.365
std	7.4525	12.7079	5.9388	14.6003	17.067
min	1.81	25.36	992.89	25.56	420.26
25%	13.51	41.74	1,009.1	63.3275	439.75
50%	20.345	52.08	1,012.94	74.975	451.55
75%	25.72	66.54	1,017.26	84.83	468.43
max	37.11	81.56	1,033.3	100.16	495.76

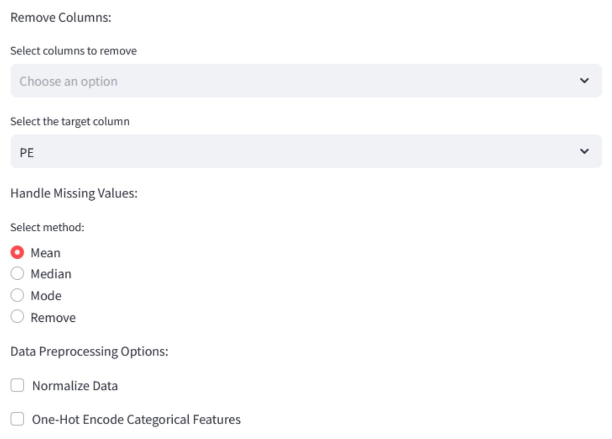
Dataset Info:

	Missing Values	Data Types
AT	0	float64
V	0	float64
AP	0	float64
RH	0	float64
PE	0	float64

Fig. 2. Dataset Preview

C. Dataset Preprocessing

Data preprocessing is a critical step to ensure the dataset is ready for modeling. In this section, users can select the features and target variable for their task. The app provides options for handling missing values, such as imputation or removal of missing data. Furthermore, users can apply normalization techniques or use one-hot encoding for categorical variables to prepare the dataset for modeling. These preprocessing options ensure that the data is clean and appropriately formatted for machine learning algorithms.



Remove Columns:

Select columns to remove: Choose an option

Select the target column: PE

Handle Missing Values:

Select method:
- [✓ Mean]
- Median
- Mode
- Remove

Data Preprocessing Options:
- ☐ Normalize Data
- ☐ One-Hot Encode Categorical Features

Fig. 3. Data Preprocessing Options

D. Choosing Task

The app automatically detects the task type—whether it is a regression or classification problem—based on the dataset. However, in cases where the task is incorrectly identified, users can manually adjust the task type to ensure the correct models and evaluation metrics are applied. This feature provides flexibility, enabling users to handle different types of problems with ease.

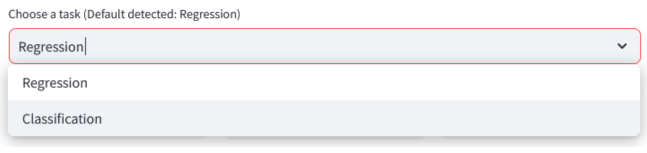
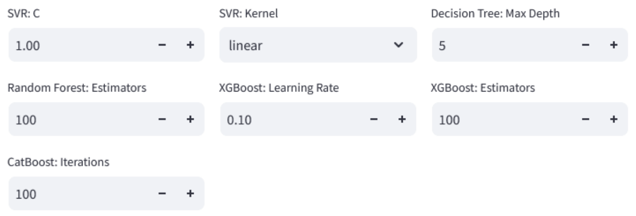


Fig. 4. Task Selection Interface

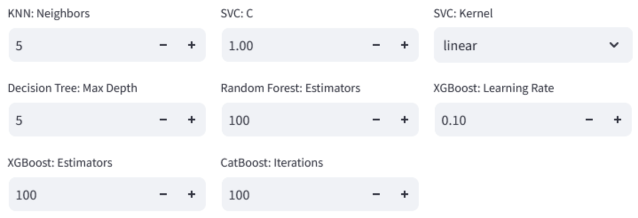
E. Hyperparameter Tuning

Hyperparameter tuning allows users to optimize the performance of machine learning models. This section enables users to fine-tune the hyperparameters for various models, such as the number of estimators in Random Forest or the learning rate in XGBoost. The app provides an interactive interface for adjusting these parameters, allowing users to experiment with different values to achieve the best model performance.



SVR: C	SVR: Kernel	Decision Tree: Max Depth
1.00	linear	5
Random Forest: Estimators	**XGBoost: Learning Rate**	**XGBoost: Estimators**
100	0.10	100
CatBoost: Iterations		
100		

Fig. 5. Hyperparameter Tuning for Regression

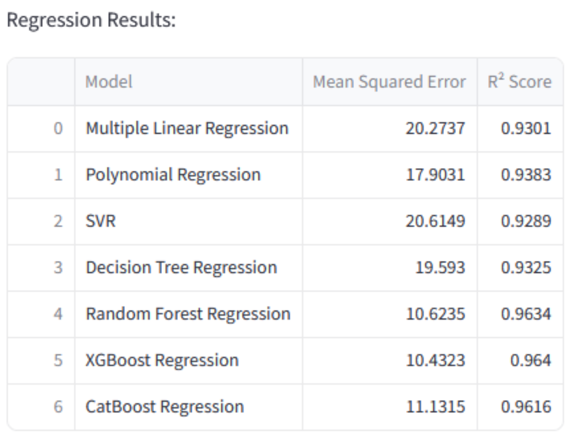


KNN: Neighbors	SVC: C	SVC: Kernel
5	1.00	linear
Decision Tree: Max Depth	**Random Forest: Estimators**	**XGBoost: Learning Rate**
5	100	0.10
XGBoost: Estimators	**CatBoost: Iterations**	
100	100	

Fig. 6. Hyperparameter Tuning for Classification

F. Results

Once the model is trained and tuned, the app generates results for various models. These results include evaluation metrics such as accuracy, precision, recall, mean squared error and r2 score, depending on the task type. The app compares the performance of different models on the dataset, providing users with insights into which model performs best for their specific use case.



Regression Results:

	Model	Mean Squared Error	R^2 Score
0	Multiple Linear Regression	20.2737	0.9301
1	Polynomial Regression	17.9031	0.9383
2	SVR	20.6149	0.9289
3	Decision Tree Regression	19.593	0.9325
4	Random Forest Regression	10.6235	0.9634
5	XGBoost Regression	10.4323	0.964
6	CatBoost Regression	11.1315	0.9616

Fig. 7. Model Performance Results

V. Results

In this section, we compare the R² scores of different regression models when implemented separately in Jupyter Notebook and when applied through the web application. The R² score is used to evaluate how well the model fits the data, with higher values indicating better model performance.

Model	R² Score (Jupyter Notebook)	R² Score (Web Application)
Multiple Linear Regression	0.9325	0.9301
Polynomial Regression	0.9458	0.9383
Support Vector Regression	0.9481	0.9289
Decision Tree Regression	0.9229	0.9325
Random Forest Regression	0.9616	0.9638

TABLE I
Comparison of R² Scores from Jupyter Notebook and Web Application

VI. Conclusion and Future Work

In this project, we have developed a user-friendly web application designed to simplify the process of model selection and evaluation for machine learning tasks. The web app integrates various preprocessing techniques and machine learning models, allowing users to upload datasets, preprocess data, and evaluate both regression and classification models interactively. By leveraging the Streamlit framework and Python libraries such as scikit-learn, XGBoost, and CatBoost, we have created an accessible platform for individuals, even those with limited technical expertise, to explore and select the most suitable machine learning models for their datasets.

The app was tested with several regression models, and the results from the web application were found to closely match those obtained from Jupyter Notebook implementations, demonstrating its reliability and accuracy. Key features of the app include dataset uploading, feature selection, missing value handling, normalization, one-hot encoding, and hyperparameter tuning, which collectively guide users through the machine learning pipeline in a streamlined manner.

Future Work

While the current version of the application focuses on regression and classification tasks, there is significant potential for extending its functionality. Future improvements could include the incorporation of unsupervised learning models, such as clustering and dimensionality reduction techniques, to handle tasks like anomaly detection and data visualization. Additionally, the application could be expanded to support reinforcement learning models for real-time decision-making problems.

Another promising direction for future work is the integration of artificial neural networks (ANNs), convolutional neural networks (CNNs), and other deep learning models. These models would significantly enhance the app's ability to tackle more complex tasks, such as image classification, natural language processing, and time-series forecasting. By incorporating deep learning frameworks like TensorFlow and PyTorch, the app could offer users a broader range of tools for more advanced machine learning applications.

The aim is to create a comprehensive, no-code solution that not only guides users through model selection but also enables them to handle more sophisticated machine learning tasks without requiring in-depth knowledge of the underlying algorithms.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Sir Dr. Khurram Jadoon and Sir Engr. Asim Shah for their invaluable guidance and support throughout the course of this project. Their expertise, patience, and dedication in teaching and mentoring us have been instrumental in helping us successfully implement this work. We greatly appreciate their continuous encouragement and for always being available to answer our questions and provide insightful feedback.