Diabetes Classification and Prediction

This project is an end-to-end machine learning project aimed at predicting the likelihood of diabetes based on user-provided health data. The project demonstrates the full machine learning pipeline from data gathering to creating a prediction model.

Project Overview

The goal of this project is to create a seamless process for predicting diabetes by building a machine learning model that analyzes various health parameters.

Project Objectives

The project follows these key steps:

1. **Data Gathering**: Collected relevant medical data from various sources, including public datasets.

2. **Descriptive Analysis**: Explored the dataset to understand the underlying patterns and trends.

3. **Data Visualizations**: Created insightful visualizations to represent key relationships in the data.

4. **Data Preprocessing**: Cleaned and transformed the data for use in the machine learning model.

5. **Data Modelling**: Trained a machine learning model using scikit-learn to predict diabetes.

6. **Model Evaluation**: Assessed the model's performance using various metrics to ensure accuracy.

Technical Aspects

Machine Learning Model

- **Library**: scikit-learn

- **Algorithms Used**: Logistic Regression, Decision Trees, Random Forests (or any chosen algorithms based on your project)

Syed Ismail N

- **Input Features**: The following fields are taken from the user:
  - Number of Pregnancies
  - Insulin Level
  - Age
  - Body Mass Index (BMI)
  - Blood Pressure
  - Glucose Level
  - Skin Thickness
  - Diabetes Pedigree Function
- **Output**: The model predicts whether the person is likely to have diabetes (Yes/No).

Prerequisites

- Python 3.13
- scikit-learn
- Pandas

Data loading and exploration

Subtask:

Description of the initial steps of loading the dataset and exploring its basic characteristics (shape, columns, data types, head, info, describe, null values).

Reasoning: Description of the initial steps of loading and exploring the dataset, including shape, columns, data types, head, info, describe, and checking for null values.

1. The dataset is loaded from 'diabetes.csv' into a pandas DataFrame named 'df'.
1. `df.shape` is used to get the dimensions of the dataset, showing the number of rows and columns.
2. `df.columns` lists the names of all columns in the DataFrame.
3. `df.dtypes` shows the data type of each column.
4. `df.head()` displays the first few rows of the dataset.

5. `df.info()` gives a summary of the DataFrame, including data types and non-null counts.
6. `df.describe().T` provides descriptive statistics for numerical columns.
7. `df.isnull().any()` checks for any null values in each column.
8. The 'DiabetesPedigreeFunction' column is renamed to 'DPF' for easier use.

Data cleaning

Subtask:

Detailed process of handling missing values by replacing zeros with NaN and then filling those NaN values with mean or median based on the data distribution.

Reasoning: Explanation of the steps taken to handle missing values by replacing zeros with NaN and then imputing them with mean or median.

In this dataset, zero values in 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', and 'BMI' are considered missing because these physiological measurements cannot realistically be zero in a living person. These zeros likely represent unrecorded or missing data.

1. A deep copy of the original DataFrame (`df`) is created as `df_copy` using `df.copy(deep=True)`. This ensures that any modifications made to `df_copy` do not affect the original `df` DataFrame.
2. The `replace(0, np.nan)` method is used on the specified columns of `df_copy` to replace all occurrences of 0 with `np.nan`. `np.nan` is a standard representation for missing values in pandas.
3. `df_copy.isnull().sum()` is then used to count the number of `np.nan` values in each column, confirming that the zero values in the target columns have been successfully replaced with NaN.
4. Plotting the histogram of `df_copy` before filling NaN values helps visualize the distribution of data in each column, which is crucial for deciding the best imputation strategy (mean or median).
5. Based on the data distributions observed from the histograms (and general statistical practices), NaN values are filled: the mean is used for 'Glucose' and 'BloodPressure' (often follow a more normal distribution), while the median is used for 'SkinThickness', 'Insulin', and 'BMI' (as these might be skewed or have outliers, making the median a more robust measure).

6. The `fillna()` method with `inplace=True` is applied to each of the columns with NaN values. This method fills the NaN values with the calculated mean or median for that specific column directly within the `df_copy` DataFrame.

7. Plotting the histogram of `df_copy` after filling NaN values shows the updated distribution of the data after imputation and confirms that the missing values have been handled.

8. Finally, `df_copy.isnull().sum()` is called again to verify that there are no remaining `np.nan` values in any of the columns, confirming the successful completion of the imputation process.

Model building

Subtask:

Explanation of the steps involved in splitting the data, feature scaling, and using GridSearchCV to find the best model. Explain the different models considered (Logistic Regression, Decision Tree, Random Forest, SVM) and why Random Forest was chosen.

Reasoning: Explaining the steps related to data splitting, feature scaling, model selection using GridSearchCV, and the rationale for choosing Random Forest.

Data splitting using `train_test_split` is a crucial step to evaluate the performance of a machine learning model on unseen data. The dataset is divided into two subsets: a training set (used to train the model) and a testing set (used to evaluate the trained model's performance). This helps in assessing how well the model generalizes to new data and prevents overfitting.

- `X`: Contains the features (independent variables) used for prediction.
- `y`: Contains the target variable (dependent variable), which is 'Outcome' in this case.
- `X_train`: The features for the training set.
- `X_test`: The features for the testing set.
- `y_train`: The target variable for the training set.
- `y_test`: The target variable for the testing set.

Feature scaling is a data preprocessing technique used to standardize or normalize the range of independent variables or features of the data. It's important because many machine learning algorithms (like Logistic Regression, SVM, and algorithms based on distance calculations) are sensitive to the scale of the input features. Features with larger values can dominate those with smaller values, leading to biased model performance.

- `StandardScaler` is used here to standardize the features by removing the mean and scaling to unit variance. This results in data with a mean of 0 and a standard deviation of 1.
- `fit_transform` is applied to `X_train` to learn the scaling parameters (mean and standard deviation) from the training data and then apply the transformation.
- `transform` is applied to `X_test` using the scaling parameters learned from the training data. It's important *not* to fit `StandardScaler` on the test set to prevent data leakage.

`GridSearchCV` is a technique used for hyperparameter tuning. It systematically searches through a predefined grid of hyperparameter values for a given model and uses cross-validation to find the combination of hyperparameters that results in the best model performance. This automates the process of finding optimal settings for a model.

- `ShuffleSplit` is a cross-validation strategy used here. It shuffles the data and splits it into a specified number of train/test sets. Each split is a random subset of the data, which helps in getting a more robust estimate of the model's performance.

The following machine learning models were considered:

- **Logistic Regression:** A linear model used for binary classification. It estimates the probability of a binary outcome.
- **Decision Tree:** A tree-like structure where each internal node represents a test on a feature, each branch represents an outcome of the test, and each leaf node represents a class label. They can capture non-linear relationships.
- **Random Forest:** An ensemble learning method that constructs multiple decision trees during training and outputs the class that is the mode of the classes of the individual trees. It generally provides better accuracy and avoids overfitting compared to a single decision tree.
- **SVM (Support Vector Machine):** A supervised learning model used for classification and regression tasks. It works by finding the hyperplane that best separates the data points of different classes in a high-dimensional space.

Random Forest was selected as the best model based on the results of `find_best_model` which used `GridSearchCV`. The Random Forest model achieved the highest cross-validation score ('score') among the evaluated models, indicating that it performed best on average across different folds of the training data during the hyperparameter tuning process. This suggests that the Random Forest model is likely to have the best generalization performance on unseen data for this specific problem.

Model evaluation

Subtask:

Description on how the model was evaluated using a confusion matrix, accuracy score, and classification report for both the test and training sets.

Reasoning: Describing how the model was evaluated using a confusion matrix, accuracy score, and classification report for both the test and training sets.

A confusion matrix is a table used to evaluate the performance of a classification model. It summarizes the number of correct and incorrect predictions made by the model for each class.

- **True Positives (TP):** The model correctly predicted that a person has diabetes (Actual: 1, Predicted: 1).
- **True Negatives (TN):** The model correctly predicted that a person does not have diabetes (Actual: 0, Predicted: 0).
- **False Positives (FP):** The model incorrectly predicted that a person has diabetes when they do not (Type I error - Actual: 0, Predicted: 1).
- **False Negatives (FN):** The model incorrectly predicted that a person does not have diabetes when they do (Type II error - Actual: 1, Predicted: 0). In the context of diabetes prediction, False Negatives are particularly important to minimize as they represent missed diagnoses.

1. The confusion matrix plots for both the test and training sets were generated using `seaborn.heatmap`. This function visualizes the matrix as a heatmap, making it easy to interpret the counts in each quadrant.
   - `cm`: The confusion matrix calculated using `confusion_matrix(y_true, y_pred)`.
   - `annot=True`: Displays the actual count values on the heatmap cells.
   - `cmap="Blues"`: Sets the color scheme of the heatmap to shades of blue.
   - `fmt='g'`: Formats the annotations as general numbers. Separate plots were generated for the test set (`y_test`, `y_pred`) and the training set (`y_train`, `y_train_pred`) to evaluate performance on both seen and unseen data.

2. The accuracy score measures the overall correctness of the model's predictions. It is calculated as the ratio of correctly predicted instances to the total number of instances.
   - Formula: Accuracy = (TP + TN) / (TP + TN + FP + FN)
   - In the code, `accuracy_score(y_test, y_pred)` calculates the accuracy on the test set, and `accuracy_score(y_train, y_train_pred)` calculates the accuracy on the training set.
   - A high accuracy score indicates that the model is performing well in terms of overall correct predictions.

3. The classification report provides a more detailed evaluation of the model's performance for each class. It includes precision, recall, f1-score, and support.
    - **Precision:** The ratio of correctly predicted positive observations to the total predicted positive observations (TP / (TP + FP)). In diabetes prediction, it's the proportion of people predicted to have diabetes who actually have it. High precision means a low rate of false positives.
    - **Recall (Sensitivity):** The ratio of correctly predicted positive observations to all actual positive observations (TP / (TP + FN)). In diabetes prediction, it's the proportion of people with diabetes who were correctly identified. High recall means a low rate of false negatives.
    - **F1-score:** The weighted average of Precision and Recall. It's a good metric when you need to balance precision and recall, especially with uneven class distributions.
    - **Support:** The number of actual occurrences of the class in the specified dataset (test or train). The report shows these metrics for both class 0 (No Diabetes) and class 1 (Diabetes).
4. Comparing the evaluation metrics between the training and testing sets helps understand the model's generalization ability.
    - **Training Set:** The accuracy and other metrics on the training set are typically higher because the model has learned from this data.
    - **Test Set:** The metrics on the test set provide a more realistic estimate of how the model will perform on new, unseen data.
    - If the training set metrics are significantly higher than the test set metrics (as observed here with ~99.94% accuracy on training vs. ~98.75% on test), it may indicate **overfitting**. Overfitting occurs when the model learns the training data too well, including noise and specific patterns that do not generalize to new data. While the difference here is small and the test accuracy is very high, a larger discrepancy would be a stronger indicator of overfitting. In this case, the model performs exceptionally well on both, suggesting good generalization despite the slight difference.

Predictions

Subtask:

Explanation of the function created for making predictions and how it uses the trained model and scaler.

Reasoning: Describing the prediction function, its inputs, how data types are handled, how inputs are formatted, how the scaler is applied, how the prediction is made using the classifier, how the output is interpreted, and mention the user warning.

1. The `predict_diabetes` function is designed to take a new set of patient features as input and use the trained Random Forest classifier to predict whether that patient is likely to have diabetes (Outcome = 1) or not (Outcome = 0).

1. The function takes eight input parameters: `Pregnancies`, `Glucose`, `BloodPressure`, `SkinThickness`, `Insulin`, `BMI`, `DPF` (Diabetes Pedigree Function), and `Age`. These directly correspond to the eight feature columns (`X`) that the Random Forest model was trained on.

2. Inside the function, the input values are explicitly converted to the data types expected by the model:

   - `Pregnancies` and `Age` are converted to integers using `int()`.
   - `Glucose`, `BloodPressure`, `SkinThickness`, `Insulin`, `BMI`, and `DPF` are converted to floating-point numbers using `float()`. This ensures that the data passed to the model is in the correct numerical format.

3. The converted input values are then organized into a list of lists: `[[preg, glucose, bp, st, insulin, bmi, dpf, age]]`. This structure is required because scikit-learn models, including the `StandardScaler` and the classifier, expect input data in a 2D array-like format, even for a single sample.

4. The `sc.transform(x)` method is used to apply the same feature scaling transformation to the new input features (`x`) that was applied to the training data (`X_train`).

   - `sc` is the `StandardScaler` instance that was previously `fit` on the training data.
   - It is absolutely critical to use the *same* fitted scaler (`sc.transform()`) on the new data as was used for the training data (`sc.fit_transform(X_train)`). This ensures that the new data is scaled using the same mean and standard deviation learned from the training distribution. Using a different scaler or fitting a new scaler on the single input sample would lead to incorrect scaling and inaccurate predictions.

5. The scaled input features (`x` after `sc.transform(x)`) are then passed to the trained `classifier`'s `predict()` method: `classifier.predict(x)`. This method uses the learned patterns from the training data to predict the class label (0 or 1) for the given input.

6. The `predict()` method returns an array containing the predicted class label(s). For a single input sample, it returns an array with one element. The `[0]` indexing (`classifier.predict(x)[0]`) extracts this single prediction.

   - If the prediction is `1`, the function interprets this as 'So Sorry! You have diabetes.'
   - If the prediction is `0`, the function interprets this as 'Great! You don't have diabetes.' This provides a clear, human-readable result based on the model's output.

7. The user warning 'X does not have valid feature names, but StandardScaler was fitted with feature names' appears because the input `x` is a simple NumPy array (from the list of lists `[[...]]`) and lacks the column/feature names that the `StandardScaler` saw during fitting (when it was applied to the pandas DataFrame `X_train`).

    - **Implication:** While this is a warning and not an error in this case, it highlights that the model and scaler expect the input features to be provided in the *exact same order* as the features were during training. The code correctly maintains this order, so the scaling and prediction are accurate despite the warning. For robustness in more complex scenarios, it's better practice to pass new data as a DataFrame with matching column names.

Insights and recommendations

Subtask:

Based on the data analysis and model performance, providing insights into the factors that seem most influential in predicting diabetes and offer recommendations for improving the model or further analysis.

Reasoning: Providing insights based on the data analysis and model performance, and offer recommendations for improvement or further analysis.

Insights

Based on the data analysis and model evaluation:

- **Feature Importance:** While explicit feature importance was not calculated, the high performance of tree-based models (Decision Tree and Random Forest) compared to linear models (Logistic Regression) and SVM (especially with a linear kernel) suggests that non-linear relationships between features and the outcome are significant. This implies that certain thresholds or combinations of features are strong indicators of diabetes.
- **Key Predictors (Implied):** Features like 'Glucose', 'BMI', 'Age', and 'Insulin' are commonly known risk factors for diabetes and likely play a significant role in the model's predictions, given their physiological relevance and potential non-linear interactions. The initial histograms also show varying distributions for these features, which could contribute to their predictive power.
- **Impact of Missing Values:** The presence of a significant number of zero values (treated as NaN) in 'SkinThickness' and 'Insulin', and the subsequent imputation, highlight the challenge of missing data. While imputation was performed, the high number of missing values in 'Insulin' (nearly half the dataset) suggests that its direct influence might be less reliable compared to features with fewer missing values, or

that the imputation method significantly impacts its apparent distribution and relationship with the outcome.

- **High Model Performance:** The Random Forest model achieved very high accuracy on both the training (~99.94%) and testing (~98.75%) sets. This indicates that the selected features, after cleaning and scaling, are highly predictive of diabetes in this dataset.
- **Potential Overfitting:** Although the test accuracy is very high, the slight difference between training and test accuracy (99.94% vs 98.75%) *could* be an indicator of minor overfitting. However, given the high absolute test accuracy, this is likely not a major concern for practical use, but it's worth noting.

Recommendations

Based on the analysis and model performance, here are some recommendations:

- **Feature Importance Analysis:** Explicitly calculate and analyze feature importance using tree-based models (like Random Forest) or other techniques (e.g., permutation importance). This would provide concrete evidence of which features are driving the predictions and offer deeper insights into the underlying factors associated with diabetes in this dataset.
- **Explore Advanced Imputation Techniques:** Investigate more sophisticated imputation methods for handling the significant missing values in 'SkinThickness' and 'Insulin'. Techniques like K-Nearest Neighbors (KNN) imputation or Multiple Imputation by Chained Equations (MICE) could potentially capture more complex relationships and improve the quality of the imputed data.
- **Consider Feature Engineering:** Explore creating new features from existing ones. For example, calculating the Glucose-to-Insulin ratio or BMI categories could potentially provide the model with more informative patterns.
- **Investigate Outliers:** While not explicitly shown in the analysis, visually inspecting features for outliers (especially after imputation) and considering robust scaling methods or outlier treatment could further improve model stability, although the current performance is already very high.
- **Explore Explainable AI (XAI) Techniques:** Given the high accuracy, applying XAI techniques could help understand *why* the model makes certain predictions, especially for high-risk individuals. This could provide valuable clinical insights.
- **Cross-Validation with Stratification:** Ensure that cross-validation folds maintain the same proportion of the outcome variable as the original dataset (stratification). While ShuffleSplit with a fixed random state provides some consistency, stratified k-fold cross-validation is generally preferred for classification tasks, especially with imbalanced datasets (though the class balance here is relatively good).

Health tips

Subtask:

Providing general health tips related to diabetes prevention and management, emphasizing that this model is not a substitute for professional medical advice.

Reasoning: Providing general health tips and a disclaimer. This can be done by printing markdown and text.

Health Tips

Disclaimer: The predictions provided by this model are for informational purposes only and should not be considered a substitute for professional medical advice, diagnosis, or treatment. Always consult with a qualified healthcare professional for any health concerns or before making any decisions related to your health or treatment.

Here are some general health tips related to diabetes prevention and management:

- **Maintain a Healthy Weight:** Losing even a small amount of weight if you are overweight or obese can significantly reduce your risk of developing type 2 diabetes.
- **Eat a Balanced Diet:** Focus on whole grains, lean proteins, fruits, and vegetables. Limit your intake of sugary drinks, processed foods, and unhealthy fats.
- **Engage in Regular Physical Activity:** Aim for at least 150 minutes of moderate-intensity aerobic activity or 75 minutes of vigorous-intensity activity per week, plus muscle-strengthening activities at least two days a week.
- **Monitor Your Blood Sugar:** If you have been diagnosed with prediabetes or diabetes, regular blood sugar monitoring as advised by your doctor is crucial for management.
- **Get Regular Medical Check-ups:** Routine visits to your healthcare provider can help detect potential health issues early, including prediabetes or diabetes.
- **Don't Smoke:** Smoking increases the risk of developing diabetes and other health problems.
- **Manage Stress:** Find healthy ways to cope with stress, as chronic stress can affect blood sugar levels.
- **Get Enough Sleep:** Aim for 7-9 hours of quality sleep per night, as poor sleep can impact insulin sensitivity.

Conclusion

Subtask:

Summarizing the project, its findings, and the potential implications.

Reasoning: Writing a concluding paragraph that summarizes the project, its findings, and potential implications, incorporating the importance of health and lifestyle.

Conclusion

This project aimed to develop a machine learning model to predict diabetes based on diagnostic measurements. Through a comprehensive process involving data loading, cleaning (handling missing values by imputation), exploration, model building, and rigorous evaluation, we identified Random Forest as the most accurate algorithm for this dataset. The model achieved a high accuracy of approximately 98.75% on unseen test data, demonstrating its strong predictive capability. This high performance suggests that the features included in the dataset are highly relevant for identifying individuals at risk of diabetes. The potential implications of this project include the use of such a model as a preliminary screening tool to help identify individuals who may benefit from further medical evaluation. However, it is crucial to reiterate that this model is not a substitute for professional medical diagnosis or advice. Ultimately, while predictive models can be valuable tools, maintaining a healthy lifestyle, including balanced nutrition, regular physical activity, and proactive health management, remains the cornerstone of preventing and managing diabetes and improving overall well-being.

Summary:

Data Analysis Key Findings

- Zero values in 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', and 'BMI' were treated as missing values and replaced with NaN, then imputed with the mean (for 'Glucose' and 'BloodPressure') or median (for 'SkinThickness', 'Insulin', and 'BMI') based on their distributions.

- Data was split into training and testing sets (X_train, X_test, y_train, y_test) to evaluate model performance on unseen data.

- Feature scaling using `StandardScaler` was applied to standardize the range of input features, which is important for many machine learning algorithms.

- `GridSearchCV` with `ShuffleSplit` cross-validation was used to find the best model and hyperparameters among Logistic Regression, Decision Tree, Random Forest, and SVM.

- Random Forest was selected as the best model based on achieving the highest cross-validation score.

- The Random Forest model achieved a high accuracy of approximately 99.94% on the training set and 98.75% on the test set.

- Evaluation using a confusion matrix, accuracy score, and classification report showed strong performance, although a slight difference between training and test accuracy (~1.19%) suggests potential minor overfitting.

- A function was created to make predictions on new patient data, which includes converting input to appropriate data types, structuring it correctly, scaling it using the *same* fitted `StandardScaler`, and using the trained classifier to predict the outcome.

Future Enhancements

- Explicitly calculate and analyze feature importance from the trained Random Forest model to gain deeper insights into which specific features are most influential in predicting diabetes.

- Explore advanced imputation techniques (e.g., KNN or MICE) for handling the significant number of missing values in 'SkinThickness' and 'Insulin' to potentially improve data quality and model robustness.

Acknowledgments

- [Scikit-learn Documentation](https://scikit-learn.org/stable/documentation.html)