**Model Experimentation and Packaging**

1. **Objective**

Train a machine learning model, perform hyperparameter tuning, and package the model for deployment.

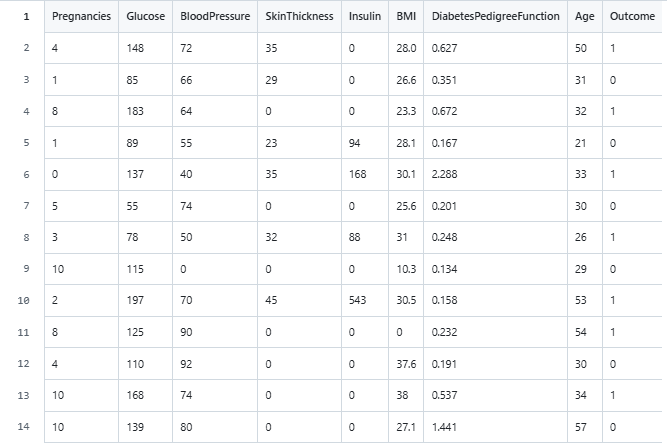
1. **Model Details**

The model we have developed is to diagnose whether a patient has diabetes or not. This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases.

### **Dataset**

The original dataset has more than 769 records. The dataset contains nine columns.

* Pregnancies: Number of times pregnant
* Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
* BloodPressure: Diastolic blood pressure (mm Hg)
* SkinThickness: Triceps skin fold thickness (mm)
* Insulin: 2-Hour serum insulin (mu U/ml)
* BMI: Body mass index (weight in kg/(height in m)^2)
* DiabetesPedigreeFunction: Diabetes pedigree function
* Age: Age (years)
* Outcome: Class variable (0 or 1)



1. **Model Preparation Process**

For this project RandomForestClassifier machine learning algorithm is used and it provided by the **scikit-learn** library that is used for classification tasks. It is an ensemble method that builds multiple decision trees during training and combines their predictions to improve accuracy and prevent overfitting. The algorithm works well for both binary and multiclass classification problems and can handle large datasets with high dimensionality. It is robust to outliers and missing values, as it uses a voting mechanism across trees. Additionally, it provides feature importance scores, making it useful for understanding the significance of input features in prediction tasks.

A screen shot of a computer program

Description automatically generated

This param\_grid dictionary defines the hyperparameter combinations for tuning a **Random Forest Classifier** using GridSearchCV. The n\_estimators key specifies the number of trees in the forest, with possible values of 100 or 200. The max\_depth key controls the maximum depth of each tree, allowing 5, 10, or unrestricted (None) as options. The min\_samples\_split key determines the minimum number of samples required to split an internal node, tested for 2 and 5. Lastly, the min\_samples\_leaf key sets the minimum number of samples that must be present in a leaf node, with values of 1 or 2.

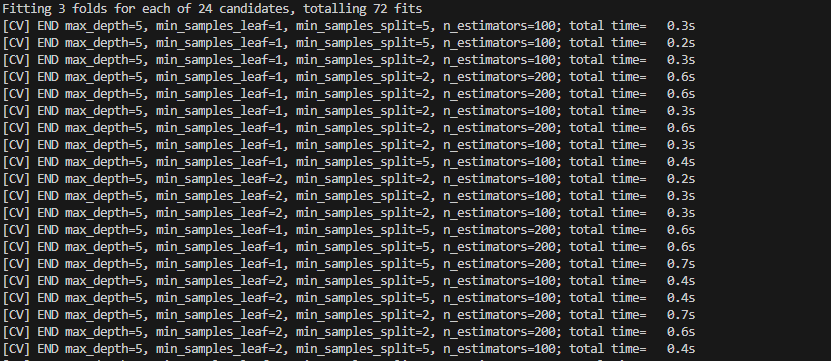
A black screen with white text

Description automatically generated

This code performs hyperparameter tuning for a Random Forest classifier using **GridSearchCV**, a technique for exhaustive search over a specified parameter grid. The param\_grid defines the hyperparameters to be optimized, such as the number of trees, maximum depth, or split criteria, for the classifier. The training data (X\_train and y\_train) is split into 3 folds for cross-validation (cv=3), and multiple configurations are evaluated in parallel (n\_jobs=-1) for efficiency. Finally, the grid\_search.fit() method trains and evaluates models with each combination of hyperparameters, returning the best-performing configuration for use.

**Hyper parameter tuning results**

* The hyperparameter tuning process evaluates 24 different combinations of parameters defined in the grid, using 3-fold cross-validation for each combination, resulting in a total of 72 fits.
* Each fit trains the model on a subset of the data and validates it on the remaining portion, ensuring that the model's performance is assessed comprehensively.
* The best combination of hyperparameters is selected based on the performance metrics from these 72 evaluations, optimizing the model for better accuracy or other specified metrics



Accuracy on best model.



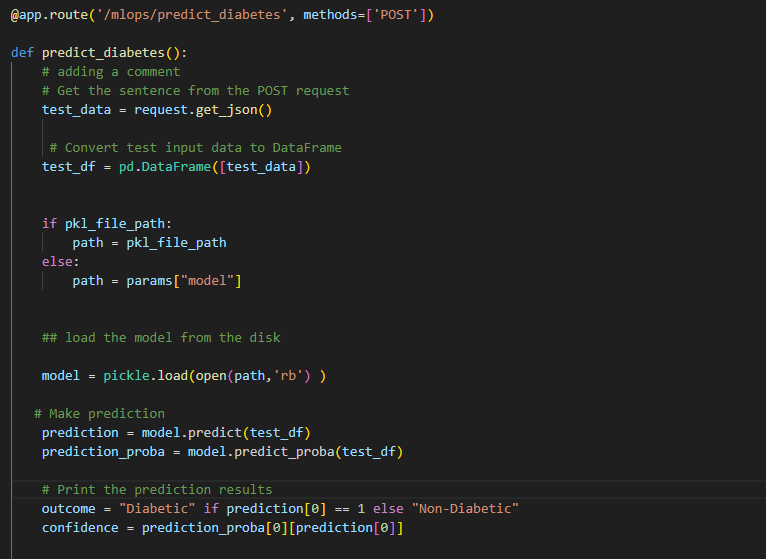
1. **Model Packaging**

Saved the machine learning model to a file, the pickle package is used to serialize the model object, enabling it to be stored and later loaded for prediction. After training a model with data, pickle.dump() is used to write the model object to a file in binary mode, which allows it to be easily transferred between systems or preserved for future use.

A black screen with white text

Description automatically generated

This same model in pkl file is loaded in the flask API file to consume externally.



The flask API file is available in the location

<https://github.com/biju123/MLopstest/blob/main/src/app.py>

1. **Docker**

The docker file definition which pack the model and the flask application is as follows

A screen shot of a computer

Description automatically generated

Once the docker file is executed by the github action, image of the application will be created and pushed to dockerhub.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated