Bank Note Authenticator Challenge

About Contest:

Context

You are provided with a dataset containing features extracted from images of banknotes. Each sample is labeled as either authentic (0) or counterfeit (1). Your task is to build a machine learning model that can accurately classify new banknote samples as either authentic or counterfeit based on the provided features.

Dataset Description:

The dataset, known as the "Bank Note Authentication UCI Data," contains 1,372 instances with four input features and one output variable (class label). The features are extracted from images of banknotes, and the class label indicates whether the banknote is authentic or counterfeit. The features included are as follows:

Variance: Variance of Wavelet Transformed image (continuous)

Skewness: Skewness of Wavelet Transformed image (continuous)

Curtosis: Curtosis of Wavelet Transformed image (continuous)

Entropy: Entropy of image (continuous)

Class: Class label (0 for authentic, 1 for counterfeit)

The dataset can be accessed from the following link: Bank Note Authentication UCI Data

Your goal is to train a classification model using this dataset and evaluate its performance on unseen test samples.

Requirements:

Load and Preprocess the Dataset:

Perform any necessary preprocessing steps, such as handling missing values, scaling the features, or encoding categorical variables (if any).

Split the Dataset:

- Split the dataset into training and testing sets. Use a reasonable ratio, such as 80% for training and 20% for testing.
- Randomly shuffle the data before splitting to ensure unbiased distribution.

Implement and Train a Classification Model:

- Choose a suitable classification algorithm, such as Logistic Regression, Decision Trees, Random Forests, Support Vector Machines, or Neural Networks.
- Implement the chosen model using a machine learning library such as scikit-learn or TensorFlow.
- **Train the model using the training data.**

Evaluate Model Performance:

- Evaluate the trained model's performance on the testing data.
- Use appropriate evaluation metrics such as accuracy, precision, recall, F1-score, or area under the ROC curve to assess the model's performance.
- Generate a classification report and confusion matrix to analyze the model's performance in more detail.

Perform Additional Analysis and Improvement:

- Perform feature engineering or feature selection techniques to improve the model's performance.
- Experiment with different hyperparameters or model architectures to achieve better results.

- Conduct a cross-validation analysis to assess the model's generalization ability.

Submission Guidelines:

For completing this challenge, the following **two files** needs to be submitted:

- A Jupyter notebook file (i.e., .ipynb file) used for code implementation. Name the file as ***geekschallange_Firstname_Lastname.ipynb***. For example, if the participant's name is John Smith, then the submission file will be named as ***geekschallange_John_Smith.ipynb***

- The trained machine learning model used for the data challenges as a pickle file [follow the appendix for better understanding]. Name the file as ***geekschallange_Firstname_Lastname.pickle***. For example, if the participant's name is John Smith, then the submission file will be named as ***geekschallange_John_Smith.pickle***

  Make sure to follow the following guidelines for naming the datafile. Do not zip these files into one zip archive, submit two independent files.

We will only accept Jupyter Notebook submissions. Please make sure that the code is properly commented, and the jupyter notebook should include the reporting about the following:

- A brief description of the data set and the problem.

- Details of the data exploration and pre-processing steps.

- Analysis of the exploration plots and insights.

- An explanation of the model selection process and the chosen model.

- Final model evaluation metrics and a discussion of model performance.

Prepare a Report:

- Summarize your approach, including the chosen model, preprocessing techniques, and any additional analysis performed.
- Include the evaluation metrics, classification report, and confusion matrix to showcase the model's performance.
- Document any insights gained from the analysis, highlighting significant findings or challenges faced.

Evaluation Criteria:

Submissions will be evaluated based on the following criteria:

- Data Pre-processing techniques
- Data exploration techniques and analysis
- Model selection and performance
- Final model evaluation and discussion
- Coding architecture and commenting
- Report Formation

Resources:

- Details of the data exploration and pre-processing steps:

https://www.geeksforgeeks.org/data-preprocessing-in-data-mining/

- Analysis of the exploration plots and insights:

https://www.geeksforgeeks.org/exploratory-data-analysis-in-python/

- An explanation of the model selection process and the chosen model:

https://www.geeksforgeeks.org/model-selection-in-machine-learning/

- Final model evaluation metrics and a discussion of model performance:

https://www.geeksforgeeks.org/evaluation-metrics-in-machine-learning/

Appendix:

What is a pickle file?

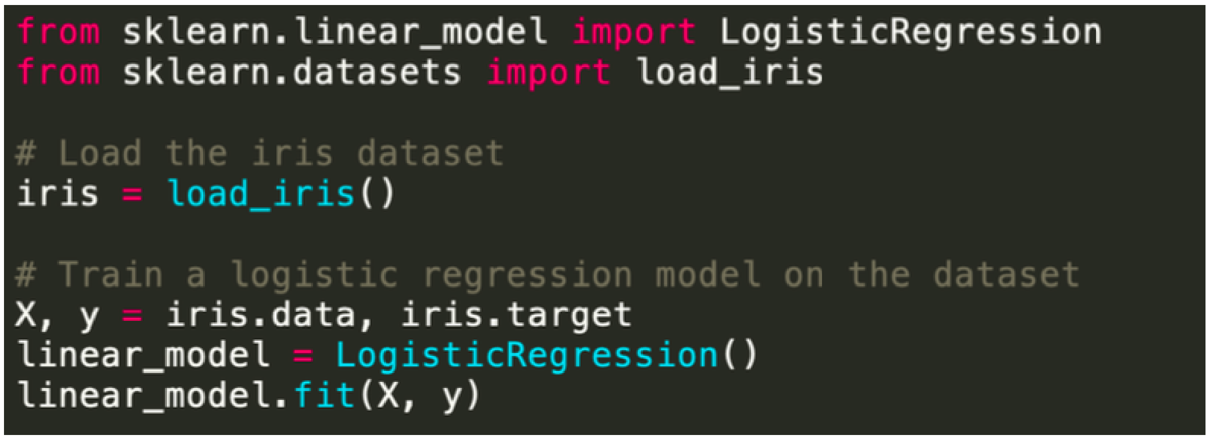
A pickle file is a serialized object that can be used to store the state of a Python object, such as a machine learning model. It is used in machine learning to save a trained model, along with any pre-processing steps that have been applied to the data, so that it can be reloaded later and used to make predictions on new data.

When you train a machine learning model, the resulting model is just a set of weights and biases that have been learned from the training data. To use the model later, you need to save these weights and biases in a file so that they can be loaded into memory and used to make predictions on new data.

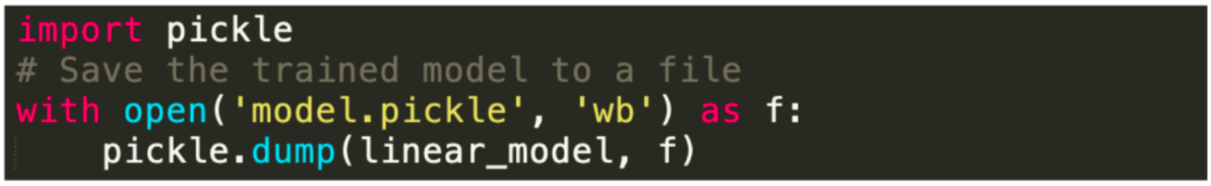
Pickle files provide a way to save Python objects to disk, so that they can be reloaded later. This is useful in machine learning because it allows you to save a trained model, along with any pre-processing steps that have been applied to the data, in a single file. This makes it easy to share and distribute trained models, and to use them in production environments.

How to save the machine learning model in a pickle file

Let’s suppose you have trained a logistic regression machine learning model on the dataset as show in the figure 2 below:

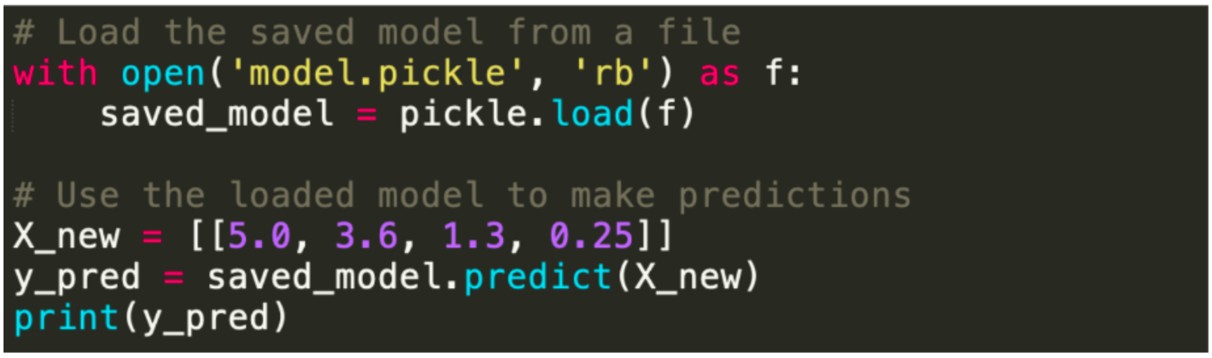


```python
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris

# Load the iris dataset
iris = load_iris()

# Train a logistic regression model on the dataset
X, y = iris.data, iris.target
linear_model = LogisticRegression()
linear_model.fit(X, y)
```



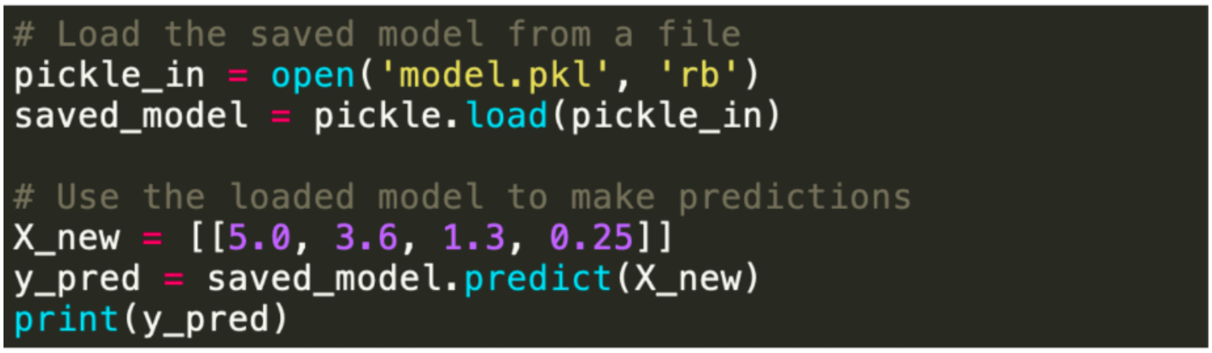
```python
import pickle
# Save the trained model to a file
with open('model.pickle', 'wb') as f:
    pickle.dump(linear_model, f)
```

To load the saved model back into memory later, you can use the pickle.load() function, like this:



```python
# Load the saved model from a file
with open('model.pickle', 'rb') as f:
    saved_model = pickle.load(f)

# Use the loaded model to make predictions
X_new = [[5.0, 3.6, 1.3, 0.25]]
y_pred = saved_model.predict(X_new)
print(y_pred)
```

Or



```python
# Load the saved model from a file
pickle_in = open('model.pkl', 'rb')
saved_model = pickle.load(pickle_in)

# Use the loaded model to make predictions
X_new = [[5.0, 3.6, 1.3, 0.25]]
y_pred = saved_model.predict(X_new)
print(y_pred)
```