Project Name: End-To-End ML with Deployment Bank Note Authentication – Random Forest – Classification

Table of Contents

```
##Dataset Link: https://www.kaggle.com/ritesaluja/bank-note-authentication-uci-data
import pandas as pd
import numpy as np
```

```
df=pd.read_csv('BankNote_Authentication.csv')
```

```
df
```

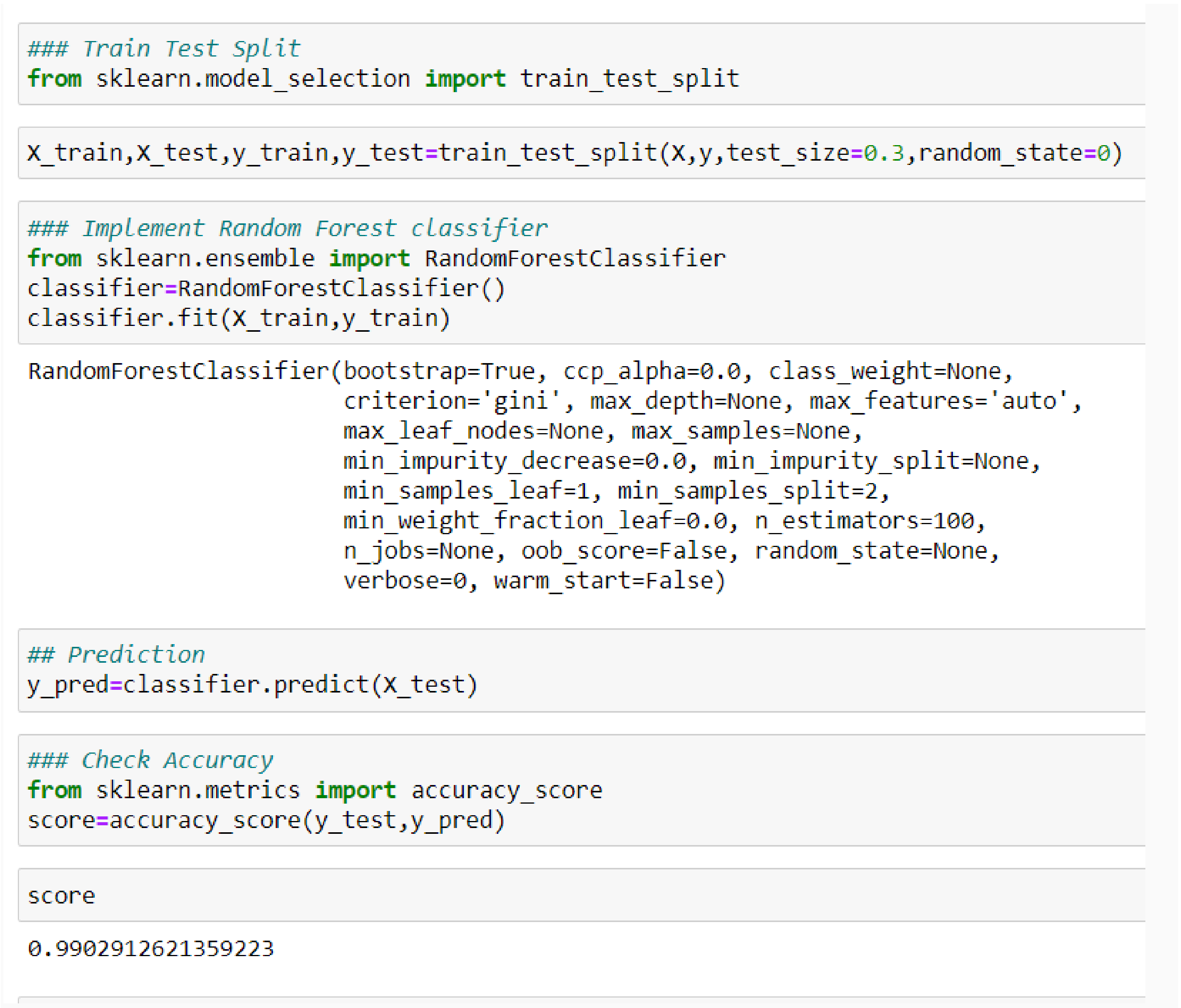
	variance	skewness	curtosis	entropy	class
0	3.62160	8.66610	-2.8073	-0.44699	0
1	4.54590	8.16740	-2.4586	-1.46210	0
2	3.86600	-2.63830	1.9242	0.10645	0
3	3.45660	9.52280	-4.0112	-3.59440	0
4	0.32924	-4.45520	4.5718	-0.98880	0
...	...	...	...	...	...
1367	0.40614	1.34920	-1.4501	-0.55949	1
1368	-1.38870	-4.87730	6.4774	0.34179	1
1369	-3.75030	-13.45860	17.5932	-2.77710	1
1370	-3.56370	-8.38270	12.3930	-1.28230	1
1371	-2.54190	-0.65804	2.6842	1.19520	1

1372 rows × 5 columns

```
### Independent and Dependent features
X=df.iloc[:,:-1]
y=df.iloc[:,-1]
```

```
X.head()
```

	variance	skewness	curtosis	entropy
0	3.62160	8.6661	-2.8073	-0.44699
1	4.54590	8.1674	-2.4586	-1.46210
2	3.86600	-2.6383	1.9242	0.10645
3	3.45660	9.5228	-4.0112	-3.59440
4	0.32924	-4.4552	4.5718	-0.98880



```
### Train Test Split
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=0)
```

```
### Implement Random Forest classifier
from sklearn.ensemble import RandomForestClassifier
classifier=RandomForestClassifier()
classifier.fit(X_train,y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

```
## Prediction
y_pred=classifier.predict(X_test)
```

```
### Check Accuracy
from sklearn.metrics import accuracy_score
score=accuracy_score(y_test,y_pred)
```

```
score
```

```
0.9902912621359223
```

Live Webapp Demo Link

https://share.streamlit.io/monicadesai-tech/project_77/main/app1.py

Deployment on Heroku: https://mlbnapp.herokuapp.com/

Abstract

The purpose of this report will be to use the BankNote_Authentication Data to classify whether it is authenticated or not.

This can be used to gain insight into how and why notes are authenticated based on certain parameters. This can also be used as a model to gain a marketing advantage, by advertisement targeting those who are more likely to perform any kind fraud with notes be it duplication and by not targeting the innocence people to help them live corruption-free world.

This is diving into Bank Note Authentication through Machine Learning Concept.

End to End Project means that it is step by step process, starts with data collection, EDA, Data Preparation which includes cleaning and transforming then selecting, training and saving ML Models, Cross-validation and Hyper-Parameter Tuning and developing web service then Deployment for end users to use it anytime and anywhere.

This repository contains the code for Bank Note Authentication using python's various libraries.
It used numpy, pandas, sklearn and pickle libraries.
These libraries help to perform individually one particular functionality.
Numpy is used for working with arrays. It stands for Numerical Python.
Pandas objects rely heavily on Numpy objects.
Matplotlib is a plotting library.
Seaborn is data visualization library based on matplotlib.
Sklearn has 100 to 200 models.
"Pickling" is the process whereby a Python object hierarchy is converted into a byte stream.
The purpose of creating this repository is to gain insights into Complete ML Project.
These python libraries raised knowledge in discovering these libraries with practical use of it.
It leads to growth in my ML repository.
These above screenshots and video in Video_File Folder will help you to understand flow of output.

Motivation

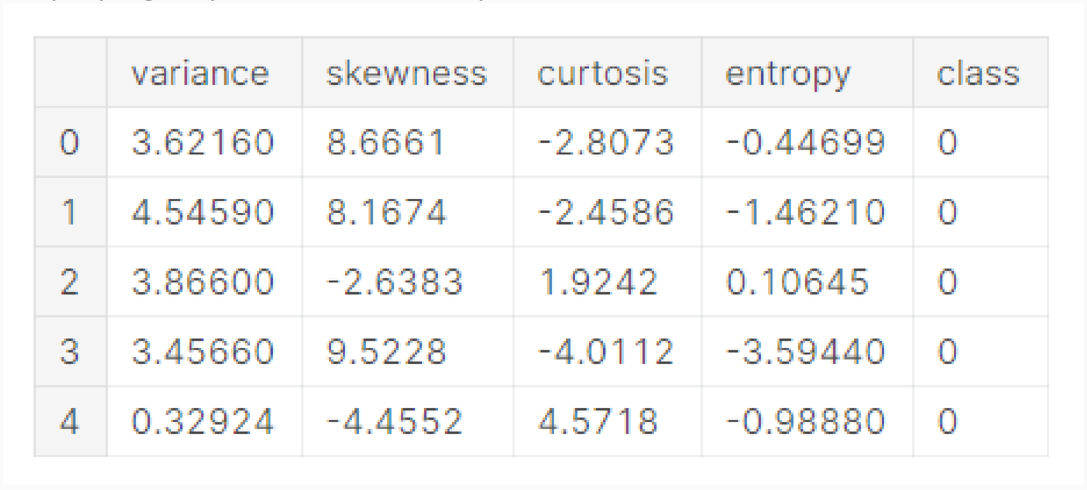
The reason behind building this project is, because many times it happens that uber driver or tuktuk driver will give you duplicate note especially in night or when you are in hurry or in dark areas. So, I can combine my knowledge of ML with the above problem to find solution to it. So, I created Bank Note Authentication Project to gain insights from IT perspective to know working of the project till making web apps in various platforms and deploying it too. Hence, I continue to spread tech wings in IT Heaven.

Acknowledgement

Dataset Available: https://archive.ics.uci.edu/ml/datasets/banknote+authentication

The Data

Displaying Dependent and Independent Features.



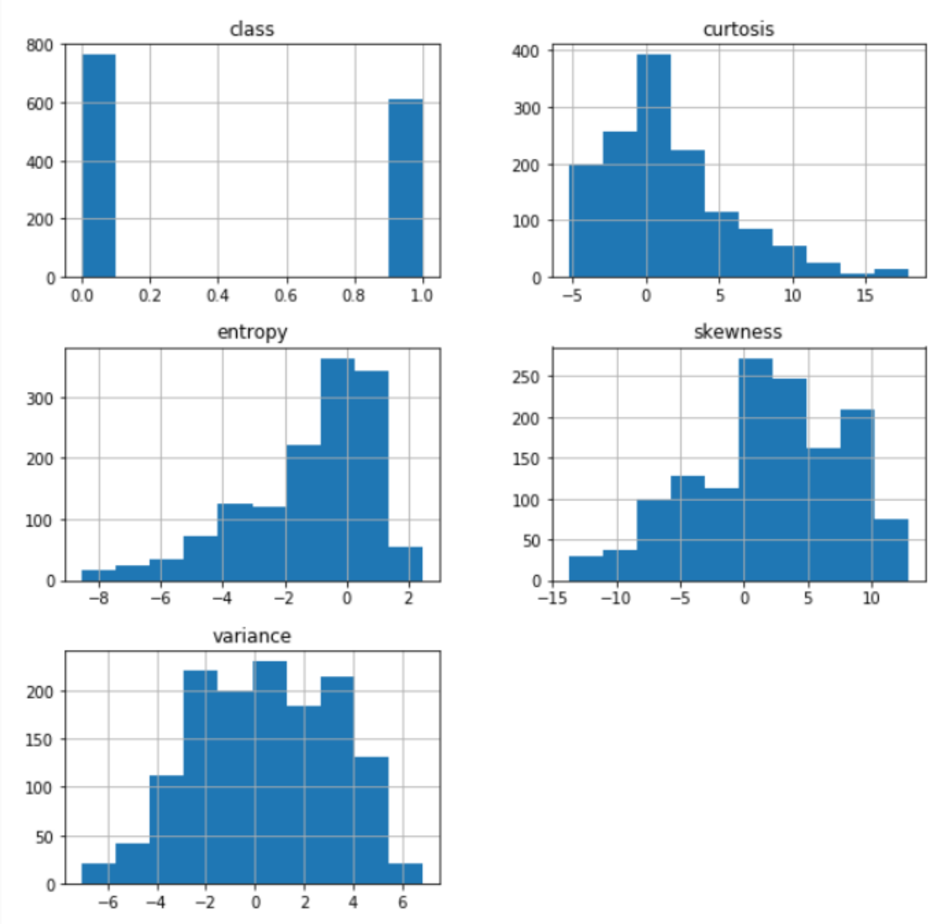
	variance	skewness	curtosis	entropy	class
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	0

There are 1372 total observations and 5 columns.
There are no missing values in this dataset.

```
Shape is : (1372, 5)
```

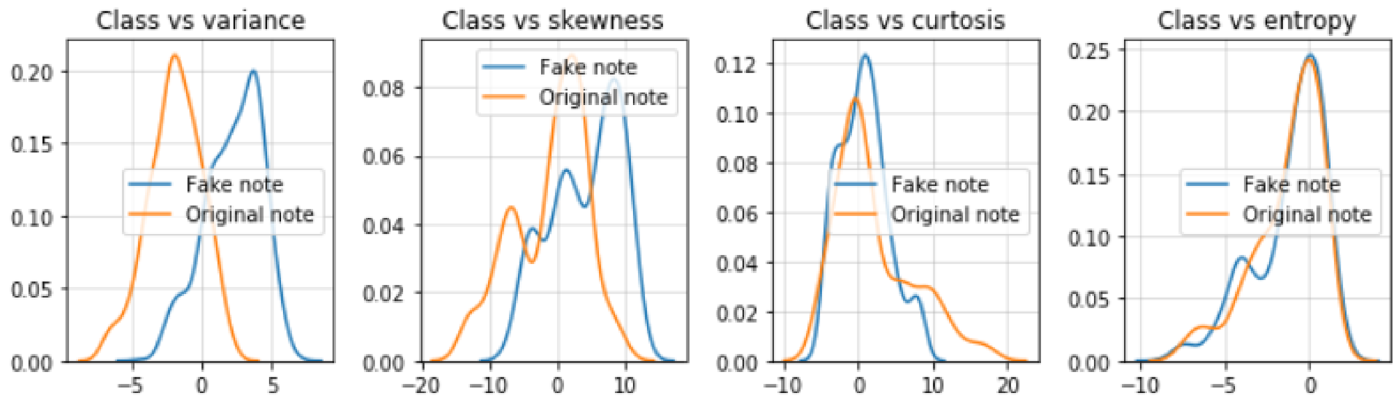
	Count	NA values	% NA	Unique	Dtype	min	25%	50%	mean	75%	max
variance	1372	0	0.0	1338	float64	-7.04	-1.77	0.50	0.43	2.82	6.82
skewness	1372	0	0.0	1256	float64	-13.77	-1.71	2.32	1.92	6.81	12.95
curtosis	1372	0	0.0	1270	float64	-5.29	-1.57	0.62	1.40	3.18	17.93
entropy	1372	0	0.0	1156	float64	-8.55	-2.41	-0.59	-1.19	0.39	2.45
class	1372	0	0.0	2	int64	0.00	0.00	0.00	0.44	1.00	1.00

Visualizing Data:



```
col_names = data.drop('class', axis = 1).columns.tolist()

plt.figure(figsize = (10,3))
i = 0
for col in col_names:
    plt.subplot(1,4,i+1)
    plt.grid(True, alpha =0.5)
    sns.kdeplot(data[col][data['class'] ==0], label = 'Fake note')
    sns.kdeplot(data[col][data['class'] ==1], label = 'Original note')
    plt.title('Class vs ' + col)
    plt.tight_layout()
    i+=1
plt.show()
```



Here are all the features included in the data set, and a short description of them all.

ColumnName	Description
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(integer)

Modelling

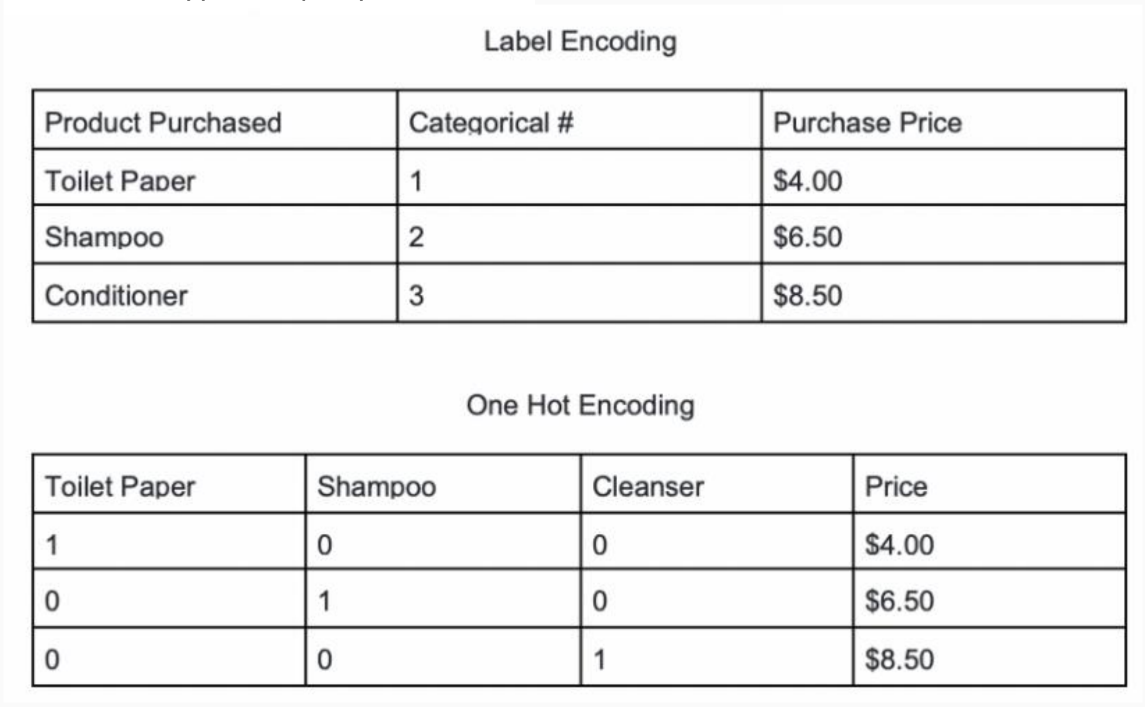
- Math behind the metrics

Where to use RF Classification Example

Our next example deals with classification data, or non-numerical data. Let's say you are doing market research for a new company who wants to know what type of people are likely to buy their products. You'll probably start by asking a sample of people in the same target market a series of questions about their buying behaviours and the kind of products they prefer. Based on their answers, you'll be able to classify them as a potential customer or not a potential customer.

Before applying the Random Forest Algorithm on these results, you will need to perform something called one-hot encoding. This entails assigning a number to a categorical variable in order to apply mathematics to the problem.

After the data is one-hot encoded, the mathematics can be applied and the Random Forest Algorithm can come to a conclusion. If the algorithm concludes that most people in this target market are not potential customers, it may be a good idea for the company to rethink their product with these types of people in mind.



Label Encoding

Product Purchased	Categorical #	Purchase Price
Toilet Paper	1	$4.00
Shampoo	2	$6.50
Conditioner	3	$8.50

One Hot Encoding

Toilet Paper	Shampoo	Cleanser	Price
1	0	0	$4.00
0	1	0	$6.50
0	0	1	$8.50

'accuarcy_score' is, in multilabel classification, this function computes subset accuracy: the set of labels predicted for a sample must *exactly* match the corresponding set of labels in y_true.
When performing Random Forests based on classification data, you should know that you are often using the Gini index, or the formula used to decide how nodes on a decision tree branch.

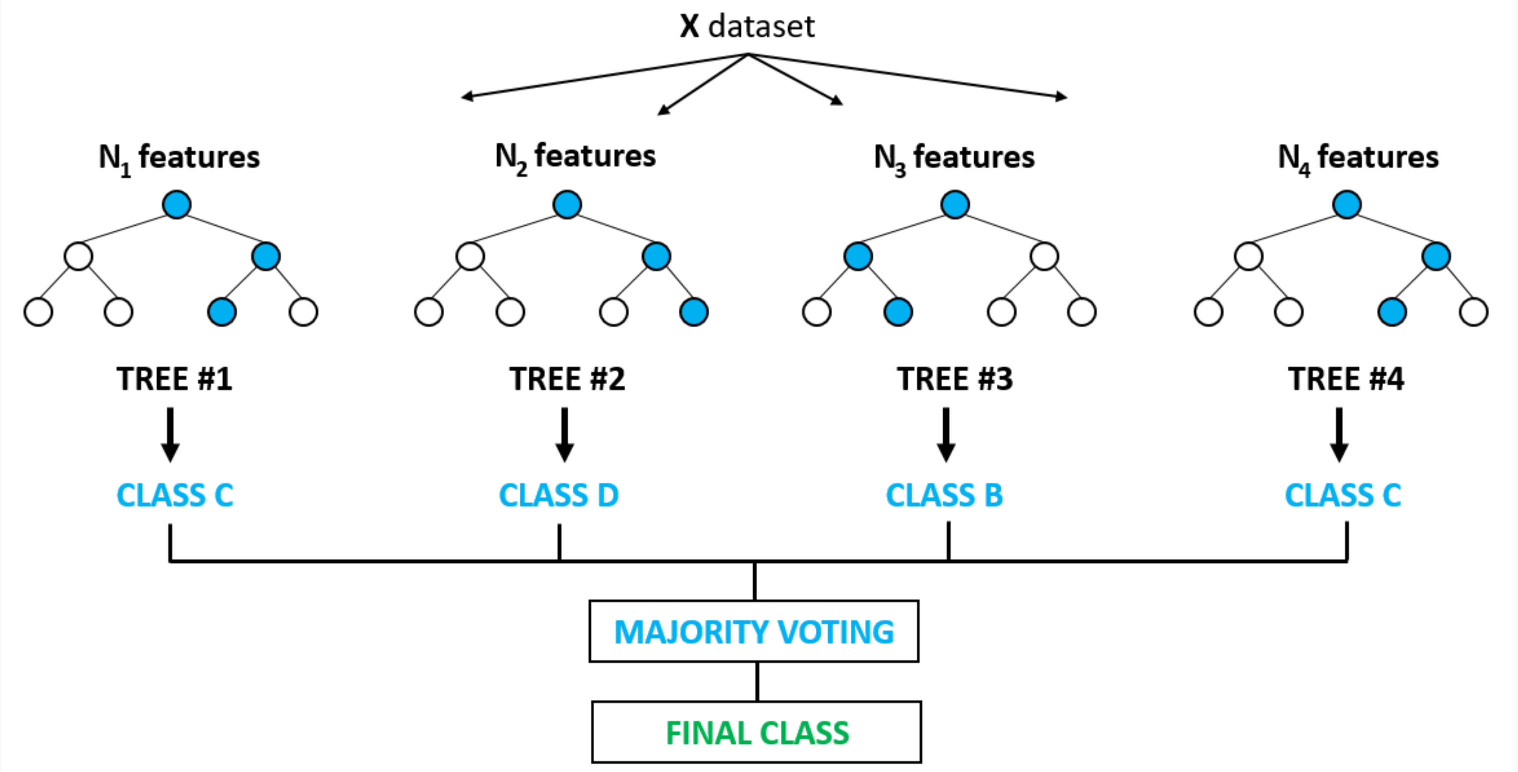
$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

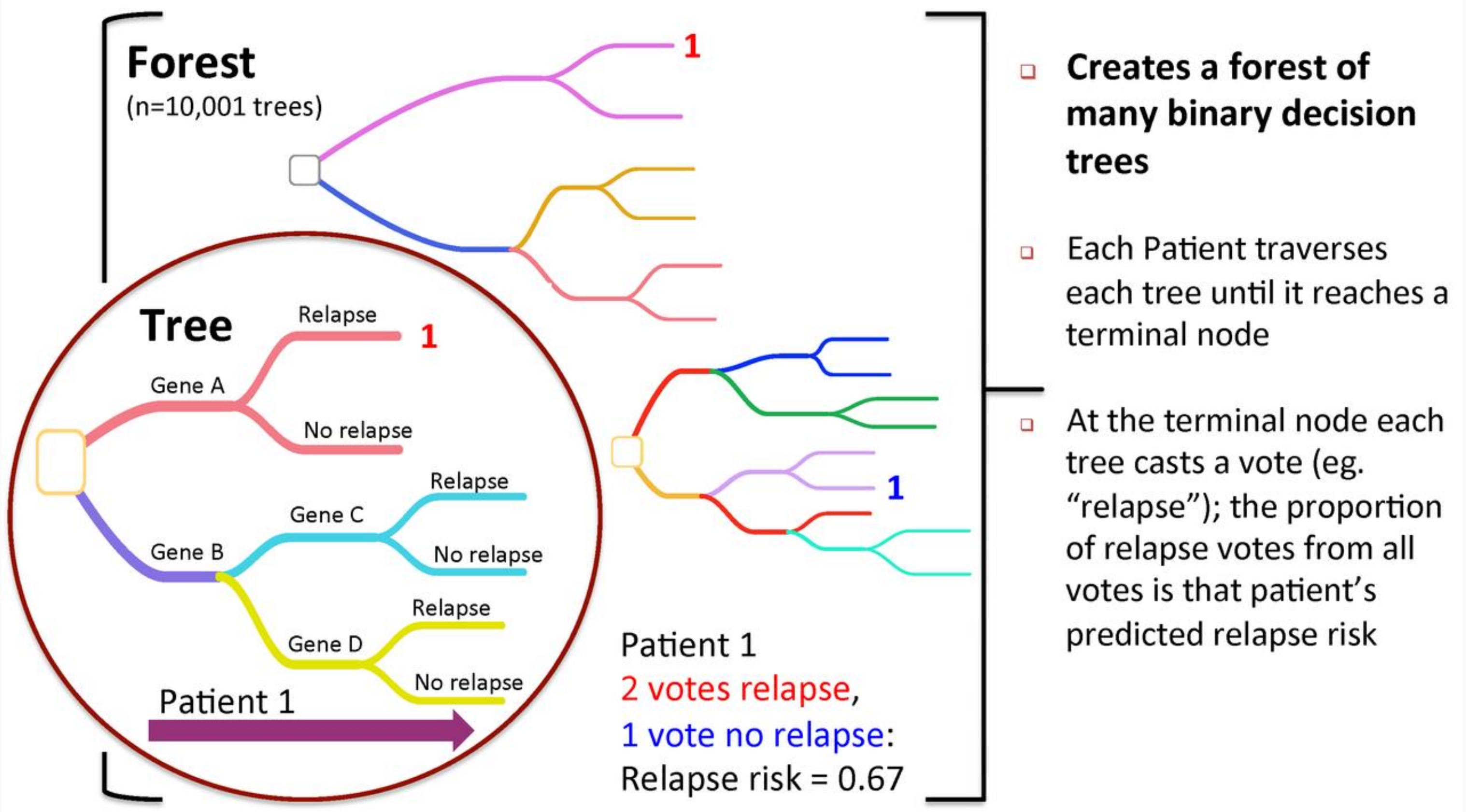
This formula uses the class and probability to determine the Gini of each branch on a node, determining which of the branches is more likely to occur. Here, *pi* represents the relative frequency of the class you are observing in the dataset and *c* represents the number of classes. You can also use entropy to determine how nodes branch in a decision tree.

$$Entropy = \sum_{i=1}^{C} -p_i * \log_2(p_i)$$

Entropy uses the probability of a certain outcome in order to make a decision on how the node should branch. Unlike the Gini index, it is more mathematical intensive due to the logarithmic function used in calculating it.

- Model Architecture Process Through Visualization





- Creates a forest of many binary decision trees
- Each Patient traverses each tree until it reaches a terminal node
- At the terminal node each tree casts a vote (eg. "relapse"); the proportion of relapse votes from all votes is that patient's predicted relapse risk

- Quick Notes

Step 1: Imported the required libraries.
Step 2: Imported the Dataset.
Step 3: Prepared Data for Training.
Step 4: Splitted the data.
Step 5: Trained the algorithm or Built the model.
Step 6: Evaluated the algorithm.
Step 7: Saved the model for re-use.
Step 8: Created web app in flask, flasgger and streamlit.

- The Model Analysis

Random forest is considered as a highly accurate and robust method because of the number of decision trees participating in the process.
It does not suffer from the overfitting problem. The main reason is that it takes the average of all the predictions, which cancels out the biases.
Random forests can also handle missing values. There are two ways to handle these: using median values to replace continuous variables, and computing the proximity-weighted average of missing values.
You can get the relative feature importance, which helps in selecting the most contributing features for the classifier.
Random forest is slow in generating predictions because it has multiple decision trees.

<u>Imported the required libraries</u> – When we import modules, we're able to call functions that are not built into Python. Some modules are installed as part of Python, and some we will install through pip. Making use of modules allows us to make our programs more robust and powerful as we're leveraging existing code.
<u>Imported and read Dataset</u> – You can import tabular data from CSV files into pandas data frame by specifying a parameter value for the file.
<u>Prepared Data for Training</u> – iloc[] method is used when the index label of a data frame is something other than numeric series of 0, 1, 2, 3…. n or in case the user doesn't know the index label. Rows can be extracted using an imaginary index position which isn't visible in the data frame. iloc stands for initial location. It performs position based selection.
<u>Splitted the data</u> – Splitting data into train and test set in order to prediction w.r.t X_test. This helps in prediction after training data.
<u>Trained the algorithm or Built the model</u> – The fit() method takes the training data as arguments, which can be one array in the case of unsupervised learning, or two arrays in the case of supervised learning. Fit function adjusts weights according to data values so that better accuracy can be achieved. It refers to how well you approximate a target function.
<u>Evaluated the algorithm</u> – By using accuracy_score and confusion_matrix. In multilabel classification, this function computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in y_true. Ground truth (correct) labels. Predicted labels, as returned by a classifier. It is the number of correct predictions made divided by the total number of predictions made, multiplied by 100 to turn it into a percentage. A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. A confusion matrix is a table that is often used to describe the

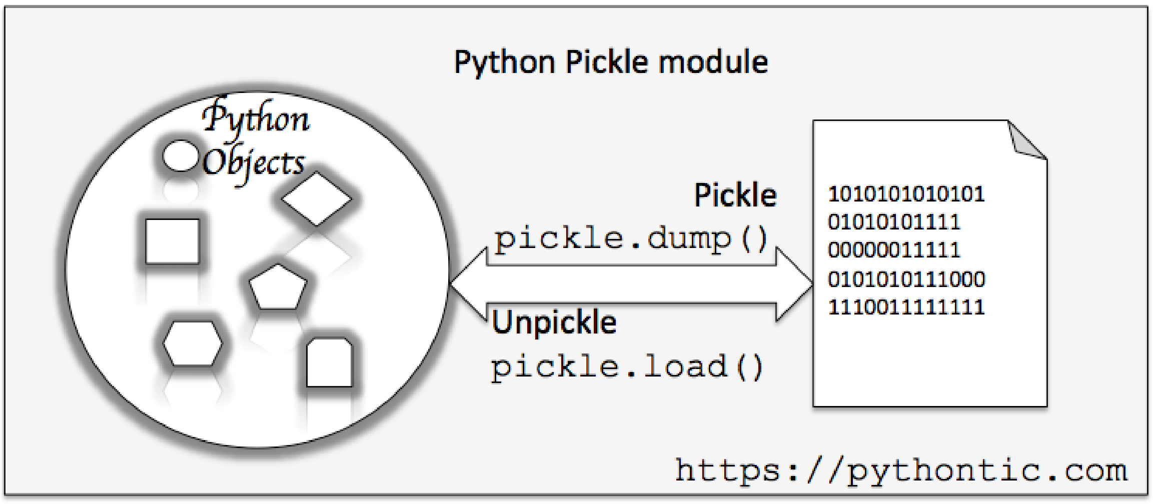
performance of a classification model (or "classifier") on a set of test data for which the true values are known.
Saved the model for re-use – Saving the created model for future reference and use so that we can directly access it rather than to go through full cycle again.
Created web app - In flask, flasgger and streamlit for end-users to use it.
Challenge that I faced while making this project was usage of flassger along with flask app.

Creation of App

Here, I am creating Flask and Flasgger App. Loading the model that we saved then passing parameters to check authentication of note.
Get input image from client then classified note and render respective .html page for solution.
Also, created it with Streamlit.

Technical Aspect

Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.
Numpy used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. It contains a multi-dimensional array and matrix data structures. It can be utilised to perform a number of mathematical operations on arrays.
Pandas module mainly works with the tabular data. It contains Data Frame and Series. Pandas is 18 to 20 times slower than Numpy. Pandas is seriously a game changer when it comes to cleaning, transforming, manipulating and analyzing data.
Sklearn is known as scikit learn. It provides many ML libraries and algorithms for it. It provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python.
Pickle in Python is primarily used in serializing and deserializing a Python object structure. In other words, it's the process of converting a Python object into a byte stream to store it in a file/database, maintain program state across sessions, or transport data over the network.



Need to train_test_split - Using the same dataset for both training and testing leaves room for miscalculations, thus increases the chances of inaccurate predictions.
The train_test_split function allows you to break a dataset with ease while pursuing an ideal model.
Also, keep in mind that your model should not be overfitting or underfitting.

Random forest is an ensemble of decision tree algorithms. The goal of ensemble methods is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator.
RF Classifier uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.
In random forests (see RandomForestClassifier and RandomForestRegressor classes), each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set. Furthermore, when splitting each node during the construction of a tree, the best split is found either from all input features or a random subset of size `max_features`.
The purpose of these two sources of randomness is to decrease the variance of the forest estimator. Indeed, individual decision trees typically exhibit high variance and tend to overfit. The injected randomness in forests yield decision trees with somewhat decoupled prediction errors. By taking an average of those predictions, some errors can cancel out. Random forests achieve a reduced variance by combining diverse trees, sometimes at the cost of a slight increase in bias. In practice the variance reduction is often significant hence yielding an overall better model.
Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.
Flasgger is a Flask extension to extract OpenAPI-Specification from all Flask views registered in your API. Flasgger also comes with SwaggerUI embedded so you can access http://localhost:5000/apidocs and visualize and interact with your API resources. Flasgger is a Flask extension to help the creation of Flask APIs with documentation and live playground powered by SwaggerUI.
Streamlit is an open-source Python library that makes it easy to create and share beautiful, custom web apps for machine learning and data science. In just a few minutes you can build and deploy powerful data apps. It is a very easy library to create a perfect dashboard by spending a little amount of time. It also comes with the inbuilt webserver and lets you deploy in the docker container. When you run the app, the localhost server will open in your browser automatically.

Installation

Using intel core i5 9th generation with NVIDIA GFORECE GTX1650.
Windows 10 Environment Used.
Already Installed Anaconda Navigator for Python 3.x
The Code is written in Python 3.8.
If you don't have Python installed then please install Anaconda Navigator from its official site.
If you are using a lower version of Python you can upgrade using the pip package, ensuring you have the latest version of pip, *python* -m pip install --*upgrade pip and press Enter*.

Run/How to Use/Steps

Keep your internet connection on while running or accessing files and throughout too.
Follow this when you want to perform from scratch.

Open Anaconda Prompt, Perform the following steps:

```
cd <PATH>
pip install flask
pip install flasgger
```

pip install streamlit
pip install numpy==1.19.3
Note: If it shows error as 'No Module Found' , then install relevant module.
You can also create requirement.txt file as, pip freeze > requirements.txt
Create Virtual Environment:
conda create -n bank_note python=3.6
y
conda activate bank_note
cd <PATH-TO-FOLDER>
run .py files.
Paste URL to browser to check whether working locally or not.

Follow this when you want to just perform on local machine.
Download ZIP File.
Right-Click on ZIP file in download section and select Extract file option, which will unzip file.
Move unzip folder to desired folder/location be it D drive or desktop etc.
Open Anaconda Prompt, write cd <PATH> and press Enter.
eg: cd C:\Users\Monica\Desktop\Projects\Python Projects 1\
23)End_To_End_Projects\Project_9_ML_FileUse_End_To_End_BankNoteAuthentication\Project_M
L_BankNoteAuth

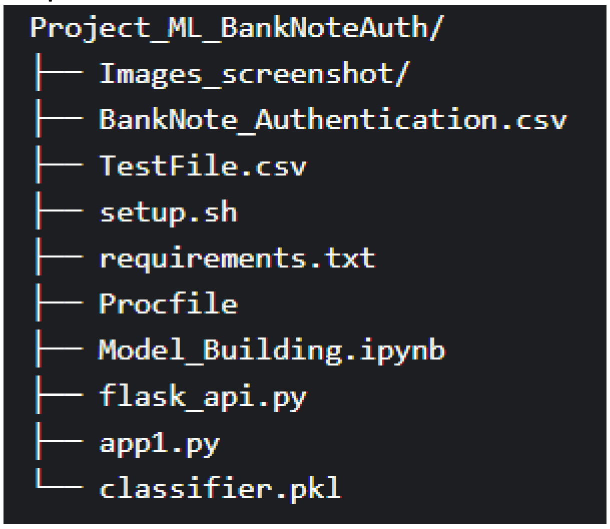
conda create -n bank_note python=3.6
y
conda activate bank_note
In Anconda Prompt, pip install -r requirements.txt to install all packages.
In Anconda Prompt, write python flask_api.py and press Enter.
Paste URL 'http://127.0.0.1:5000/apidocs' to browser to check whether working locally or not.

In Anaconda Prompt, write streamlit run app1.py and press Enter.
Paste URL (if not opened automatically) to browser and check whether working locally or not.
Please be careful with spellings or numbers while typing filename and easier is just copy filename and then run it to avoid any silly errors.
Note: cd <PATH>
[Go to Folder where file is. Select the path from top and right-click and select copy option and paste it next to cd one space <path> and press enter, then you can access all files of that folder]
[cd means change directory]

Directory Tree/Structure of Project

Folder: 23)End_To_End_Projects > Project_9_ML_FileUse_End_To_End_BankNoteAuthentication > Project_ML_BankNoteAuth

BankNote_Authentication.csv
ModelTraining.ipynb
flask_api.py
app1.py
classifier.pkl
Procfile
requirements.txt



```
Project_ML_BankNoteAuth/
├── Images_screenshot/
├── BankNote_Authentication.csv
├── TestFile.csv
├── setup.sh
├── requirements.txt
├── Procfile
├── Model_Building.ipynb
├── flask_api.py
├── app1.py
└── classifier.pkl
```

To Do/Future Scope

Can deploy on AWS and Google Cloud.

Technologies Used/System Requirements/Tech Stack

Download the Material

You can Download Dataset from here: https://github.com/monicadesAI-tech/Project_77/blob/main/BankNote_Authentication.csv

You can Download Entire Project from here: https://github.com/monicadesAI-tech/Project_77

You can Download Model Building from here: https://github.com/monicadesAI-tech/Project_77/blob/main/Model_Building.ipynb

You can Download Web App_File from here: https://github.com/monicadesAI-tech/Project_77/blob/main/flask_api.py

https://github.com/monicadesAI-tech/Project_77/blob/main/app1.py

You can see Detailed Project at Website here: https://github.com/monicadesAI-tech.github.io/banknote.html

Download dependencies from here: https://github.com/monicadesAI-tech/Project_77/blob/main/requirements.txt

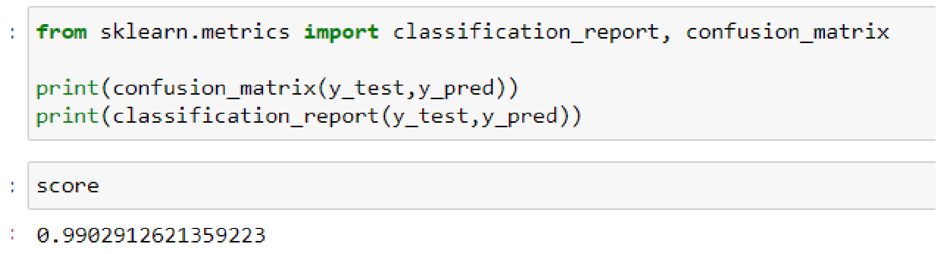
Download saved model from here: https://github.com/monicadesAI-tech/Project_77/blob/main/classifier.pkl

Conclusion

- Modelling

Can check AUC-ROC Score and Plotting graph too.

- Analysis



```
from sklearn.metrics import classification_report, confusion_matrix

print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
score
```

```
0.9902912621359223
```

Credits

Krish Naik

https://www.kaggle.com/balams/bank-note-authentication-classification

https://www.kaggle.com/manthankyada/bank-note-authentication-using-random-forest

Paper Citation

https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf