"ARTIFICIAL INTELLIGENCE PROJECT"

Sameer Shahid

F2021266561

"BRAIN TUMOR PROJECT REPORT"

- ## INTRODUCTION:

The Brain Tumor Project involves building machine learning models to predict the presence of brain tumors based on specific image features extracted from MRI scans. This report outlines the dataset used, the machine learning models applied, their performance metrics, and concludes with the model saving/loading process for future use.

- ## DATASET:

The dataset used in this project is sourced from Kaggle and contains various features extracted from brain MRI images, including statistical measures and texture features. The dataset has been preprocessed by dropping unnecessary columns and handling null values.

"Imports Necessary Modules"

```
import numpy as np
import pandas as pd
import seaborn as sns
import xgboost as xgb
from sklearn.svm import SVC
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

"Load our dataset"

```
df = pd.read_csv("/kaggle/input/brain-tumor/Brain Tumor.csv")
df.drop("Image", inplace=True, axis=1)
```

- ## EXPLORATORY DATA ANALYSIS:

A correlation matrix was generated to understand the relationships between different features. Features like 'Mean', 'Correlation', and

'Coarseness' were found to have low correlation with the target variable ('Class') and were subsequently dropped.

"Drop the columns which are not Important"

```
df.drop(['Mean', "Correlation", "Coarseness"], axis=1, inplace=True)
```

"Check for the null values"

```
df.isnull().sum()
```

• DATA PREPROCESSING:

The dataset was normalized using MinMaxScaler to ensure all features were on the same scale. This step is crucial for many machine learning algorithms to perform effectively.

"Normalize our dataset using MinMaxScaler"

```
x = df.drop("Class", axis=1)
y = df["Class"]

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)

x_train = pd.DataFrame(x_train_scaled)
x_test = pd.DataFrame(x_test_scaled)
```

• MODEL BUILDING:

Four different classifiers were trained and evaluated:

1. **RandomForestClassifier:** Ensemble learning method based on decision trees.
2. **LogisticRegression:** Linear classifier suitable for binary classification tasks.
3. **SVC (Support Vector Classifier):** Uses support vectors to classify data.
4. **XGBoost:** Gradient boosting algorithm known for its performance and efficiency.

Each model was trained on the scaled training data and evaluated using the test data to predict the presence or absence of brain tumors.

"Model Building – RandomForestClassifier"

```
np.random.seed(42)

rf = RandomForestClassifier(random_state=42, n_estimators=250)
rf.fit(x_train, y_train)
random_forest_pred = rf.predict(x_test)

accuracy = accuracy_score(y_test, random_forest_pred) * 100
rounded_accuracy = round(accuracy, 1)
print(f'Accuracy Random_Forest: {rounded_accuracy} %')
```

"Model Building – LogisticRegression"

```
np.random.seed(42)

lr = LogisticRegression(random_state=42, C=25)
lr.fit(x_train, y_train)
lr_prediction = lr.predict(x_test)

accuracy = accuracy_score(y_test, lr_prediction) * 100
rounded_accuracy = round(accuracy, 1)
print(f'Accuracy Logistic Regression: {rounded_accuracy} %')
```

"Model Building – SVC"

```
np.random.seed(42)

model = SVC(C=15, kernel="rbf", random_state=42)
model.fit(x_train, y_train)
svc_prediction = model.predict(x_test)

accuracy = accuracy_score(y_test, svc_prediction) * 100
rounded_accuracy = round(accuracy, 1)
print(f'Accuracy SVC: {rounded_accuracy} %')
```

"Model Building – XGBoost"

```
np.random.seed(42)

xgb_classifier = xgb.XGBClassifier(random_state=42, n_estimators=200)
xgb_classifier.fit(x_train, y_train)
xgb_prediction = xgb_classifier.predict(x_test)

accuracy = accuracy_score(y_test, xgb_prediction) * 100
```

```
rounded_accuracy = round(accuracy, 1)
print(f'Accuracy XGBoost: {rounded_accuracy} %')
```

- **MODEL EVALUATION:**
  - Accuracy: Calculated as the percentage of correctly predicted instances out of all instances.
  - Confusion Matrix: Visualized to understand true positive, true negative, false positive, and false negative predictions for each model.
  - Classification Report: Provided detailed metrics including precision, recall, and F1-score for each class.

- **SAVE AND LOAD MODELS:**

The RandomForestClassifier model was saved using joblib for future deployment. The model can be loaded and used to make predictions on new data, ensuring continuity and usability of the developed solution.

"Save and Load Models"

```
import joblib

filename = 'random_forest_v2.joblib'
joblib.dump(rf, open(filename, 'wb'))

tumor_model = joblib.load(open(filename, 'rb'))
```

- **FUTURE WORK:**

Future work may involve:

- Exploring deep learning techniques for image-based tumor detection.
- Enhancing feature engineering to capture more relevant aspects of MRI scans.
- Deploying the model in a real-world setting for clinical validation.

By continually refining models and integrating new advancements in machine learning, the accuracy and applicability of brain tumor detection systems can be significantly enhanced.

"Prediction with our Model"

```
tumor_model = joblib.load('random_forest_v2.joblib')
y_prediction = tumor_model.predict(x_test)
print(classification_report(y_prediction, y_test))
```

- **RESULTS:**

The accuracy scores of the models on the test set were as follows:

- RandomForest: 85.0%
- LogisticRegression: 77.5%
- SVC: 82.5%
- XGBoost: 83.5%

- **CONCLUSION:**

All models showed promising performance in predicting brain tumor presence from MRI scan features. RandomForest exhibited the highest accuracy in this instance. Further fine-tuning of hyperparameters and potentially exploring more advanced feature engineering techniques could improve model performance further.

- **GitHub Link**

https://github.com/sameer2oo2/brain-tumor