Task Report

Task Title: Breast Cancer Classification using Random Forest with Feature Selection

Objective:

The objective of this project is to classify breast cancer cases as malignant or benign using machine learning techniques. This involves:

- Feature extraction and preprocessing.
- Applying feature selection methods (Filter, Wrapper, and Embedded).
- Training a Random Forest Classifier and evaluating its performance.

Steps Performed

Step 1: Importing Libraries and Uploading the Dataset

The necessary libraries for data manipulation, visualization, and machine learning were imported. The dataset was uploaded to the Colab environment.

Code:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.feature_selection import SelectKBest, chi2, RFE
from sklearn.ensemble import ExtraTreesClassifier
from google.colab import files

uploaded = files.upload()
```

Step 2: Loading and Preprocessing the Dataset

- The dataset was read into a pandas DataFrame.
- Missing values were handled by dropping rows with `NaN` values.
- The target variable, `diagnosis`, was encoded as:
    - Malignant (M) = 1
    - Benign (B) = 0

Code:

```
df = pd.read_csv("breast-cancer.csv")
```

```
print(df.head())
print(df.info())
print(df.describe())

df = df.dropna()

# Encoding target variable: 'M' -> 1 (Malignant), 'B' -> 0 (Benign)
df['diagnosis'] = df['diagnosis'].map({'M': 1, 'B': 0})
```

Step 3: Defining Features (X) and Target (y)

The features (`X`) and target variable (`y`) were separated for further processing.

Code:

```
# Separating features and target
X = df.drop(columns=['diagnosis'])
y = df['diagnosis']
```

Step 4: Applying Feature Selection Techniques

4.1 Filter Method

The **SelectKBest** method with the Chi-Square test was used to select the top 10 features based on statistical relevance.

Code:

```
# Using Filter method: SelectKBest with chi2
filter_selector = SelectKBest(score_func=chi2, k=10)
X_selected_filter = filter_selector.fit_transform(X, y)
selected_features_filter = X.columns[filter_selector.get_support()]

print("Selected Features (Filter Method):", list(selected_features_filter))
```

4.2 Wrapper Method

The **Recursive Feature Elimination (RFE)** method was applied using Random Forest to iteratively select the most important 10 features.

Code:

```
# Using Wrapper method: Recursive Feature Elimination (RFE)
rfe_selector = RFE(estimator=RandomForestClassifier(random_state=42),
n_features_to_select=10, step=1)
rfe_selector.fit(X, y)
selected_features_wrapper = X.columns[rfe_selector.get_support()]

print("Selected Features (Wrapper Method):",
list(selected_features_wrapper))
```

4.3 Embedded Method

Feature importance was evaluated using the **ExtraTreesClassifier** to select the top 10 features.

Code:

```
# Using Embedded method: Feature importance using RandomForestClassifier
model_embedded = ExtraTreesClassifier(random_state=42)
model_embedded.fit(X, y)
importances = model_embedded.feature_importances_
selected_features_embedded = X.columns[np.argsort(importances)[-10:]]  #
Select top 10 features

print("Selected Features (Embedded Method):",
list(selected_features_embedded))
```

Step 5: Splitting the Dataset and Scaling Features

The dataset was split into training and testing sets, and features were scaled using `StandardScaler`.

Code:

```
X_selected = X[selected_features_filter]  # You can change this to wrapper
or embedded features

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X_selected, y,
test_size=0.2, random_state=42)

# Scale the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Step 6: Training the Random Forest Classifier

A Random Forest Classifier was trained on the processed data.

Code:

```
# Training Random Forest model
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluating accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy * 100:.2f}%")
```

Step 7: Saving Results

The actual and predicted results were saved to a CSV file for further analysis.

Code:

```
results = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
results.to_csv('results.csv', index=False)
print("Results saved to results.csv")
```

Model Accuracy:

The Random Forest Classifier achieved an accuracy of **94.74%.**

Conclusion

This project successfully demonstrated:

1. Preprocessing and feature extraction.
2. Application of three feature selection methods.
3. Training a Random Forest Classifier.
4. Achieving high accuracy in classifying breast cancer cases.