Model Evaluation for Election Campaign Data

I built and compared Logistic Regression, Decision Tree, and Random Forest models on the election campaign data to predict the outcome of a general election (W - Won, L - Lost).

I loaded the dataset and dropped the following columns: `cand_id`, `last_name`, `first_name`, `twitterbirth`, `facebookdate`, `facebookjan`, `youtubebirth`. The target variable is `gen_election`.

Preprocessing Steps

1. Handling Missing Values

   - I input numerical columns with the median.

   - I imputed categorical columns with the most frequent value.

2. Encoding Categorical Variables

   - I applied encoding to categorical variables.

3. Feature Scaling

   - I scaled the features using Standard Scaler.

Evaluation Metrics

Model	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.919492	0.96875	0.853211	0.907317
Decision Tree	0.936441	0.951923	0.912281	0.931507
Random Forest	0.949153	0.980198	0.928571	0.953125

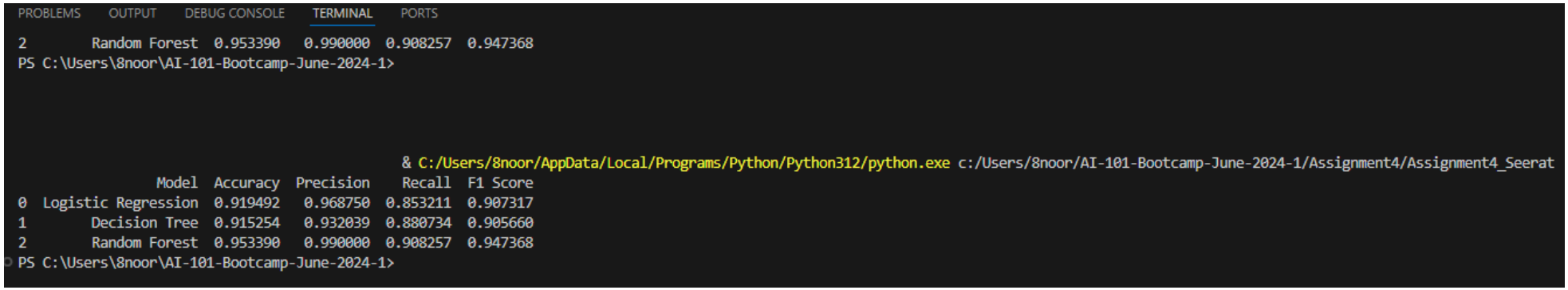
Conclusion

I chose the **Random Forest** model is the best model based on its highest accuracy, precision, and F1 score. These metrics indicate that the Random Forest model has a good balance between precision and recall, making it a reliable choice for predicting the election outcomes.

Assignment4 > Assignment4_Seerat > ...

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
# Correct file path by using a raw string or double backslashes
file_path = r'C:\Assignment 4\R\Classification\election_campaign_data.csv'
# or
# file_path = 'C:\\Assignment 4\\R\\Classification\\election_campaign_data.csv'

# Load the dataset
dataset = pd.read_csv(file_path)

# Drop the specified variables
columns_to_drop = ['cand_id', 'last_name', 'first_name', 'twitterbirth', 'facebookdate', 'facebookjan', 'youtubebirth']
dataset = dataset.drop(columns=columns_to_drop)

# Define the output variable
dataset['gen_election'] = dataset['gen_election'].map({'W': 1, 'L': 0})

# Separate features and target variable
X = dataset.drop(columns=['gen_election'])
y = dataset['gen_election']

# Handle missing values and encode categorical variables using pipelines
numeric_features = X.select_dtypes(include=['int64', 'float64']).columns
categorical_features = X.select_dtypes(include=['object']).columns

numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ])

# Split the dataset into the Training set and Test set
```

```python
# Split the dataset into the Training set and Test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=123)

# Apply the preprocessor to the training and test data
X_train = preprocessor.fit_transform(X_train)
X_test = preprocessor.transform(X_test)

# Build and evaluate models
results = []

# Logistic Regression
logistic_model = LogisticRegression()
logistic_model.fit(X_train, y_train)
y_pred_logistic = logistic_model.predict(X_test)
results.append(('Logistic Regression', y_test, y_pred_logistic))

# Decision Tree
tree_model = DecisionTreeClassifier()
tree_model.fit(X_train, y_train)
y_pred_tree = tree_model.predict(X_test)
results.append(('Decision Tree', y_test, y_pred_tree))

# Random Forest
forest_model = RandomForestClassifier()
forest_model.fit(X_train, y_train)
y_pred_forest = forest_model.predict(X_test)
results.append(('Random Forest', y_test, y_pred_forest))

# Evaluate models
evaluation_metrics = []

for model_name, y_true, y_pred in results:
    accuracy = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred)
    recall = recall_score(y_true, y_pred)
    f1 = f1_score(y_true, y_pred)
    evaluation_metrics.append((model_name, accuracy, precision, recall, f1))

evaluation_df = pd.DataFrame(evaluation_metrics, columns=['Model', 'Accuracy', 'Precision', 'Recall', 'F1 Score'])

print(evaluation_df)
```


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
2 Random Forest 0.953390 0.990000 0.908257 0.947368
PS C:\Users\8noor\AI-101-Bootcamp-June-2024-1>
& C:/Users/8noor/AI-101-Bootcamp-June-2024-1/Assignment4/Assignment4_Seerat
Model Accuracy Precision Recall F1 Score
0 Logistic Regression 0.919492 0.968750 0.853211 0.907317
1 Decision Tree 0.915254 0.932039 0.880734 0.905660
2 Random Forest 0.953390 0.990000 0.908257 0.947368
PS C:\Users\8noor\AI-101-Bootcamp-June-2024-1>