Final Classification Assignment – IPL Matches Dataset

Barnita Das (Roll Number: 06)

Q1. Use a Unique Dataset for Classification

For this assignment, I have used the **IPL Matches Dataset** (`matches.csv`). The goal is to predict whether `team1` wins a match based on pre-match conditions.

Target Variable:

- `team1_win` – 1 if `team1` wins, 0 otherwise.

Features Used:

- `team1`, `team2`, `toss_winner`, `toss_decision`, `venue`

Preprocessing Steps:

- Removed rows with missing values
- Created binary target variable
- Applied Label Encoding to categorical variables
- Performed a 60-20-20 train-validation-test split

Code:

```
df = pd.read_csv("matches.csv")
df.dropna(inplace=True)
df['team1_win'] = (df['team1'] == df['winner']).astype(int)
features = ['team1', 'team2', 'toss_winner', 'toss_decision', 'venue']
X = df[features]
y = df['team1_win']
le = LabelEncoder()
X_encoded = X.apply(le.fit_transform)
X_train, X_temp, y_train, y_temp = train_test_split(X_encoded, y,
    ↪ test_size=0.4)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp,
    ↪ test_size=0.5)
```

Q2. Select Minimum of Three Classification Algorithms

I trained the following classifiers:

- Logistic Regression
- Random Forest Classifier
- K-Nearest Neighbors (KNN)

Training Code:

```
lr = LogisticRegression(max_iter=500)
rf = RandomForestClassifier(random_state=42)
knn = KNeighborsClassifier()
lr.fit(X_train, y_train)
rf.fit(X_train, y_train)
knn.fit(X_train, y_train)
```

Q3. Choose the Best Model Based on Validation Performance

Validation Accuracy Scores:

- Logistic Regression: 0.58
- Random Forest: 0.52
- KNN: 0.50

Code:

```
val_scores = {
    "Logistic Regression": accuracy_score(y_val, lr.predict(X_val)),
    "Random Forest": accuracy_score(y_val, rf.predict(X_val)),
    "KNN": accuracy_score(y_val, knn.predict(X_val))
}
```

Hence, **Logistic Regression** was selected as the best model.

Q4. Report Accuracy of Selected Model on Test Data

Test Accuracy: 0.55

Classification Report:

	precision	recall	f1-score	support
0	0.54	0.60	0.57	111
1	0.56	0.50	0.53	109
accuracy			0.55	220

Confusion Matrix:

```
[[67 44]
 [54 55]]
```

Code:

```
y_pred_test = lr.predict(X_test)
print(accuracy_score(y_test, y_pred_test))
print(confusion_matrix(y_test, y_pred_test))
```

Q5. Ensemble Classification Approach

I implemented a **Voting Classifier** using hard voting with Logistic Regression, Random Forest, and KNN.

Code:

```
ensemble = VotingClassifier(estimators=[
    ('lr', lr), ('rf', rf), ('knn', knn)
], voting='hard')
ensemble.fit(X_train, y_train)
y_ensemble = ensemble.predict(X_test)
ensemble_acc = accuracy_score(y_test, y_ensemble)
```

Ensemble Accuracy: 0.54

Classification Report:

	precision	recall	f1-score	support
0	0.53	0.60	0.56	111
1	0.55	0.48	0.51	109
accuracy			0.54	220

Confusion Matrix:

```
[[67 44]
 [57 52]]
```

Discussion:

Why Ensemble May Perform Slightly Worse ?

- Weak classifiers like KNN reduce accuracy
- Voting ignores confidence or probabilities
- Ensemble may overfit with inconsistent models
- Poorer models can outweigh stronger ones

Data Limitations

- No player-level performance features
- Venue and toss decision have low variance
- Match date, weather, and recent form not included
- Label encoding may impose artificial order
- Class imbalance may affect learning

Summary Table

Model	Validation Accuracy	Test Accuracy
Logistic Regression	0.58	0.55
Random Forest	0.52	0.50
KNN	0.50	0.48
Ensemble (Voting)	–	0.54

Table: Model comparison based on accuracy