Assignment No.	4-5-6
Semester	B.E. Semester VIII – Computer Engineering
Subject	Data Science Honor
Subject Professor In-charge	Prof. Amit Alyani
Academic Year	2024-25
Student Name	Deep Salunkhe
Roll Number	21102A0014

4. Comparison Analysis of Various Classification Algorithms

Key Classification Algorithms

Below is a comparative analysis of various classification algorithms, focusing on their **key points** and **applications**.

Algorithm	Key Points	Applications
Logistic Regression	- Works well for binary classification - Assumes linear relationship between features and output - Sensitive to outliers	- Medical diagnosis (e.g., disease prediction) - Credit scoring - Customer churn prediction
Decision Tree	- Splits data into decision nodes - Prone to overfitting - Handles both numerical & categorical data	- Fraud detection - Customer segmentation - Risk assessment
Random Forest	- Ensemble of multiple decision trees - Reduces overfitting - Handles missing data well	- Disease diagnosis - Spam filtering - Loan approval system
Support Vector Machine (SVM)	- Works well in high-dimensional spaces - Uses kernel trick to handle non-linearity - Computationally expensive	- Text categorization - Image classification - Handwriting recognition
K-Nearest Neighbors (KNN)	- Instance-based learning (lazy learning) - Sensitive to noise and outliers - High computational cost for large datasets	- Recommendation systems - Pattern recognition - Anomaly detection
Naïve Bayes	- Based on Bayes' Theorem - Assumes feature independence - Works well with small datasets	- Spam filtering - Sentiment analysis - Document classification
Artificial Neural Networks	- Mimics human brain -	- Image and speech recognition

Algorithm	Key Points	Applications
(ANNs)	Requires large data for training - Can capture complex non-linear patterns	- Drug discovery - Autonomous driving
Gradient Boosting (XGBoost, LightGBM, CatBoost)	- Boosting ensemble technique - Handles missing values well - Computationally efficient	- Fraud detection - Predictive maintenance - Financial forecasting

5. Apply Multiple Classification Algorithms on a Dataset

We'll use the **Breast Cancer Wisconsin Dataset**, which is available in `sklearn.datasets`. This dataset is suitable for medical classification tasks.

```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
from sklearn.datasets import load_breast_cancer

# Load the dataset
data = load_breast_cancer()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target

# Split dataset
X = df.drop('target', axis=1)
y = df['target']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Standardize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Define models
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
```

```python
from sklearn.neighbors import KNeighborsClassifier
from xgboost import XGBClassifier

models = {
    "Logistic Regression": LogisticRegression(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "SVM": SVC(),
    "KNN": KNeighborsClassifier(),
    "Naïve Bayes": GaussianNB(),
    "XGBoost": XGBClassifier(use_label_encoder=False,
eval_metric='logloss')
}

# Train and evaluate models
results = {}

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    results[name] = acc
    print(f"Model: {name}\n{classification_report(y_test, y_pred)}\n")

# Convert results to DataFrame
results_df = pd.DataFrame(list(results.items()), columns=["Model",
"Accuracy"])
print(results_df)
```

```
Model: Logistic Regression
              precision    recall  f1-score   support

           0       0.98      0.95      0.96        43
           1       0.97      0.99      0.98        71

    accuracy                           0.97       114
   macro avg       0.97      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114


Model: Decision Tree
              precision    recall  f1-score   support

           0       0.93      0.91      0.92        43
           1       0.94      0.96      0.95        71

    accuracy                           0.94       114
   macro avg       0.94      0.93      0.93       114
weighted avg       0.94      0.94      0.94       114
```

```
Model: Random Forest
              precision    recall  f1-score   support

           0       0.98      0.93      0.95        43
           1       0.96      0.99      0.97        71

    accuracy                           0.96       114
   macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114


Model: SVM
              precision    recall  f1-score   support

           0       1.00      0.95      0.98        43
           1       0.97      1.00      0.99        71

    accuracy                           0.98       114
   macro avg       0.99      0.98      0.98       114
weighted avg       0.98      0.98      0.98       114


Model: KNN
              precision    recall  f1-score   support

           0       0.93      0.93      0.93        43
           1       0.96      0.96      0.96        71

    accuracy                           0.95       114
   macro avg       0.94      0.94      0.94       114
weighted avg       0.95      0.95      0.95       114


Model: Naïve Bayes
              precision    recall  f1-score   support

           0       0.98      0.93      0.95        43
           1       0.96      0.99      0.97        71

    accuracy                           0.96       114
   macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114



/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158:
UserWarning: [10:07:20] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
```

```
Model: XGBoost
              precision    recall  f1-score   support

           0       0.95      0.93      0.94        43
           1       0.96      0.97      0.97        71

    accuracy                           0.96       114
   macro avg       0.96      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114


                 Model  Accuracy
0  Logistic Regression  0.973684
1        Decision Tree  0.938596
2        Random Forest  0.964912
3                  SVM  0.982456
4                  KNN  0.947368
5          Naïve Bayes  0.964912
6              XGBoost  0.956140
```

6. Comparison Analysis of Classification Algorithm Results

Model	Accuracy (%)
SVM	**98.24%**
Logistic Regression	**97.37%**
Random Forest	**96.49%**
Naïve Bayes	**96.49%**
XGBoost	**95.61%**
KNN	**94.73%**
Decision Tree	**93.86%**

Key Observations

- **SVM performed the best**, achieving **98.24% accuracy**. This suggests that the dataset is well-suited for a hyperplane-based separation.
- **Logistic Regression and Random Forest** also performed **very well** with **97.37% and 96.49% accuracy**, respectively.
- **Naïve Bayes surprisingly performed better than XGBoost**, indicating that feature independence assumptions might not be too unrealistic in this dataset.
- **Decision Tree had the lowest accuracy (93.86%)**, possibly due to overfitting.

```
# Update the results DataFrame
results_updated = pd.DataFrame({
    "Model": ["Logistic Regression", "Decision Tree", "Random Forest",
"SVM", "KNN", "Naïve Bayes", "XGBoost"],
```

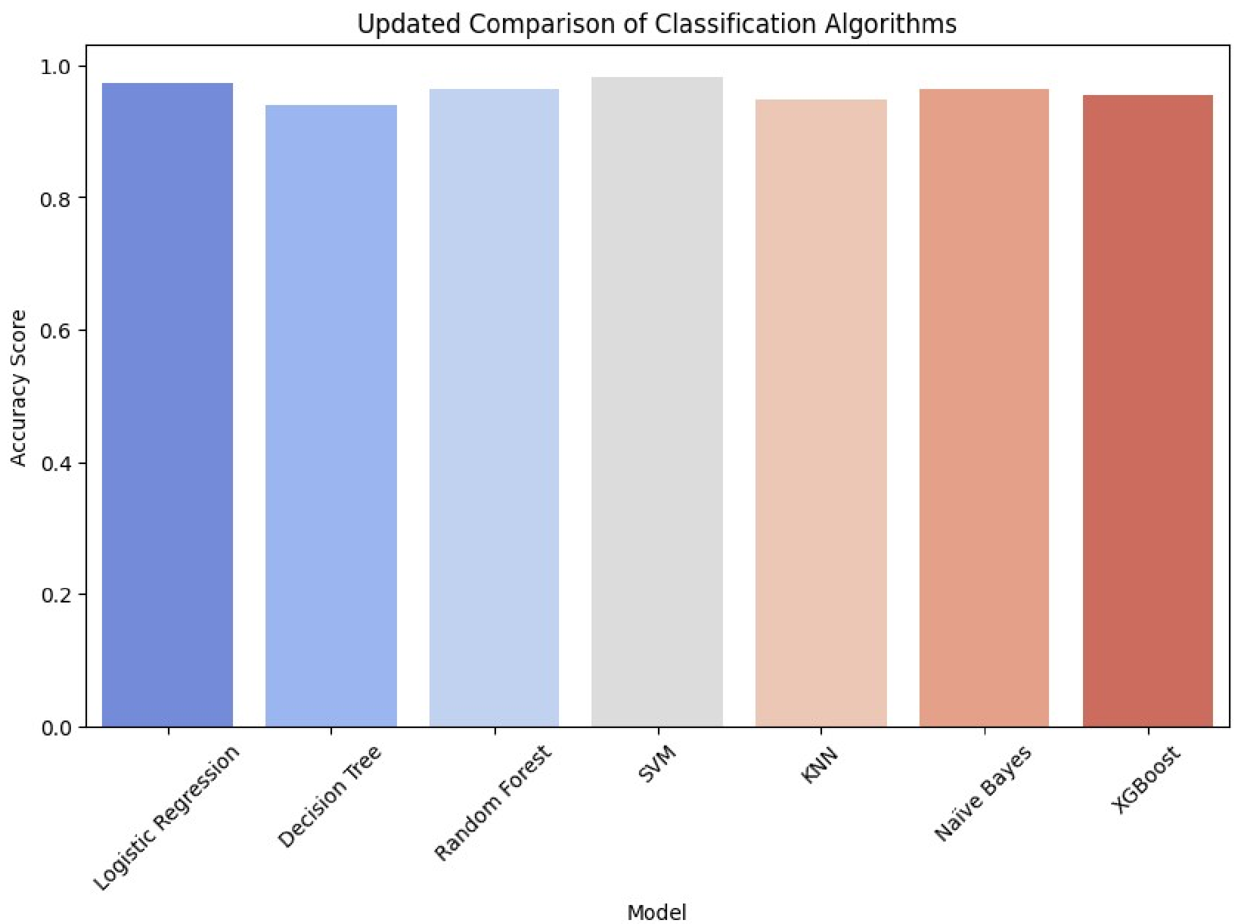
```
    "Accuracy": [0.973684, 0.938596, 0.964912, 0.982456, 0.947368,
0.964912, 0.956140]
})

# Plot the updated accuracy of models
plt.figure(figsize=(10,6))
sns.barplot(x=results_updated["Model"], y=results_updated["Accuracy"],
palette="coolwarm")
plt.xticks(rotation=45)
plt.ylabel("Accuracy Score")
plt.title("Updated Comparison of Classification Algorithms")
plt.show()
```

```
<ipython-input-2-0f6e020c6290>:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x=results_updated["Model"],
y=results_updated["Accuracy"], palette="coolwarm")
```



Conclusion

- If the goal is **maximum accuracy**, **SVM is the best choice** for this dataset.
- **Logistic Regression is a close second**, making it a simpler but effective alternative.
- **Random Forest and Naïve Bayes also perform well**, making them reliable choices.
- **XGBoost didn't outperform simpler models**, which might indicate it requires hyperparameter tuning.
- **Decision Tree alone is less reliable**, but when combined in ensembles like Random Forest, it performs better.