MAWLANA BHASHANI SCIENCE AND TECHNOLOGY UNIVERSITY

Lab Report

Department: Computer Science and Engineering

Course Title: Artificial Intelligence and Machine Learning Lab
Course Code: CSE 4102

Report Tittle: Applying various Machine learning algorithm in the dataset.

Submitted By:

Shamim Sorkar (CE19044)

Submitted To:

Lubna Yasmin Pinky

Assistant Professor,

Dept. of CSE, MBSTU.

Name of Experiment: There is a dataset given which contains the information of various users obtained from the social networking sites. There is a car making company that has recently launched a new SUV car. So, the company wanted to check how many users from the dataset, wants to purchase the car.

Introduction: In Machine Learning algorithm, we firstly train the system with train data then predict the output. Supervised learning algorithm mainly used for classification problems. We start by preprocessing the data, which involves reading the dataset, handling missing values, and scaling the features using StandardScaler. Then, we split the data into training and test sets using train_test_split. Next, we define a function to evaluate the performance of classification models based on accuracy, precision, recall, F1-score, and confusion matrix. We implement several classification algorithms such as Logistic Regression, Decision Tree, Random Forest, Support Vector Machine (SVM), and k-Nearest Neighbors (kNN).

Data Description:

User_ID	Gender	Age	EstimatedSalary	Purchased
15624510	Male	19	15000	0
15810944	Male	35	20000	0
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	0
15728773	Male	27	58000	0
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	0
15727311	Female	35	65000	0
15570769	Female	26	80000	0
15606274	Female	26	52000	0
15746139	Male	20	86000	0
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1
15617482	Male	45	26000	1
15704583	Male	46	28000	1
15621083	Female	48	29000	1
15649487	Male	45	22000	1
15736760	Female	47	49000	1

Source Code:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix

from sklearn.linear_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

from matplotlib.colors import ListedColormap


data = pd.read_csv("dataset.csv")

data.dropna(inplace=True)

X = data[['Age', 'EstimatedSalary']]

y = data['Purchased']


scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)


X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

```
def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    conf_matrix = confusion_matrix(y_test, y_pred)

    f1 = precision = recall = 1

    if conf_matrix.shape[0] < 2 or conf_matrix.shape[1] < 2:
        tp = fp = fn = tn = 0
    else:
        tp = conf_matrix[0, 0]
        fp = conf_matrix[0, 1]
        fn = conf_matrix[1, 0]
        tn = conf_matrix[1, 1]

        precision = tp / (tp + fp) if tp + fp > 0 else 0
        recall = tp / (tp + fn) if tp + fn > 0 else 0
        f1 = (2 * precision * recall) / (precision + recall)

    return accuracy, precision, recall, f1, conf_matrix

results = {}
```

```
confusion_matrices = {}

log_reg_model = LogisticRegression()
log_reg_model.fit(X_train, y_train)
log_reg_results_train = evaluate_model(log_reg_model, X_train, y_train)
log_reg_results_test = evaluate_model(log_reg_model, X_test, y_test)
results['Logistic Regression'] = log_reg_results_test[:-1]
confusion_matrices['Logistic Regression'] = log_reg_results_test[-1]

decision_tree_model = DecisionTreeClassifier()
decision_tree_model.fit(X_train, y_train)
decision_tree_results_train = evaluate_model(decision_tree_model, X_train, y_train)
decision_tree_results_test = evaluate_model(decision_tree_model, X_test, y_test)
results['Decision Tree'] = decision_tree_results_test[:-1]
confusion_matrices['Decision Tree'] = decision_tree_results_test[-1]

random_forest_model = RandomForestClassifier(n_estimators=100, random_state=42)
random_forest_model.fit(X_train, y_train)
random_forest_results_train = evaluate_model(random_forest_model, X_train, y_train)
random_forest_results_test = evaluate_model(random_forest_model, X_test, y_test)
results['Random Forest'] = random_forest_results_test[:-1]
confusion_matrices['Random Forest'] = random_forest_results_test[-1]
```

```
svm_model = SVC(kernel='linear', random_state=42)
svm_model.fit(X_train, y_train)
svm_results_train = evaluate_model(svm_model, X_train, y_train)
svm_results_test = evaluate_model(svm_model, X_test, y_test)
results['SVM'] = svm_results_test[:-1]
confusion_matrices['SVM'] = svm_results_test[-1]

knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train, y_train)
knn_results_train = evaluate_model(knn_model, X_train, y_train)
knn_results_test = evaluate_model(knn_model, X_test, y_test)
results['kNN'] = knn_results_test[:-1]
confusion_matrices['kNN'] = knn_results_test[-1]

for model_name, conf_matrix in confusion_matrices.items():
    print(f"\nConfusion Matrix for {model_name} (Test set):\n{conf_matrix}")

metrics_df = pd.DataFrame(results, index=['Accuracy', 'Precision', 'Recall', 'F1-Score']).T
print("Test Set Results:\n", metrics_df)

for model_name, model in {
    'Logistic Regression': log_reg_model,
    'Decision Tree': decision_tree_model,
```

```
    'Random Forest': random_forest_model,
    'SVM': svm_model,
    'kNN': knn_model
}.items():
    plt.figure(figsize=(12, 5))

    # Plot training set
    plt.subplot(1, 2, 1)
    x_set_train, y_set_train = X_train, y_train
    x1_train, x2_train = np.meshgrid(np.arange(start=x_set_train[:, 0].min() - 1,
stop=x_set_train[:, 0].max() + 1, step=0.01),
                                     np.arange(start=x_set_train[:, 1].min() - 1, stop=x_set_train[:, 1].max() +
1, step=0.01))
    plt.contourf(x1_train, x2_train, model.predict(np.array([x1_train.ravel(),
x2_train.ravel()]).T).reshape(x1_train.shape),
                 alpha=0.75, cmap=ListedColormap(('purple', 'green')))
    plt.xlim(x1_train.min(), x1_train.max())
    plt.ylim(x2_train.min(), x2_train.max())
    for i, j in enumerate(np.unique(y_set_train)):
        plt.scatter(x_set_train[y_set_train == j, 0], x_set_train[y_set_train == j, 1],
                    c=ListedColormap(('purple', 'green'))(i), label=j)
    plt.title(f'{model_name} (Training set)')
    plt.xlabel('Age')
    plt.ylabel('Estimated Salary')
    plt.legend()
```

```
    # Plot test set

    plt.subplot(1, 2, 2)

    x_set_test, y_set_test = X_test, y_test

    x1_test, x2_test = np.meshgrid(np.arange(start=x_set_test[:, 0].min() - 1, stop=x_set_test[:,
0].max() + 1, step=0.01),

                                   np.arange(start=x_set_test[:, 1].min() - 1, stop=x_set_test[:, 1].max() + 1,
step=0.01))

    plt.contourf(x1_test, x2_test, model.predict(np.array([x1_test.ravel(),
x2_test.ravel()]).T).reshape(x1_test.shape),

                 alpha=0.75, cmap=ListedColormap(('purple', 'green')))

    plt.xlim(x1_test.min(), x1_test.max())

    plt.ylim(x2_test.min(), x2_test.max())

    for i, j in enumerate(np.unique(y_set_test)):

        plt.scatter(x_set_test[y_set_test == j, 0], x_set_test[y_set_test == j, 1],

                    c=ListedColormap(('purple', 'green'))(i), label=j)

    plt.title(f'{model_name} (Test set)')

    plt.xlabel('Age')

    plt.ylabel('Estimated Salary')

    plt.legend()



    plt.show()
```

Output:

Confusion Matrix for Logistic Regression (Test set):

[[5]]

Confusion Matrix for Decision Tree (Test set):

[[3 2]

 [0 0]]

Confusion Matrix for Random Forest (Test set):

[[3 2]

 [0 0]]

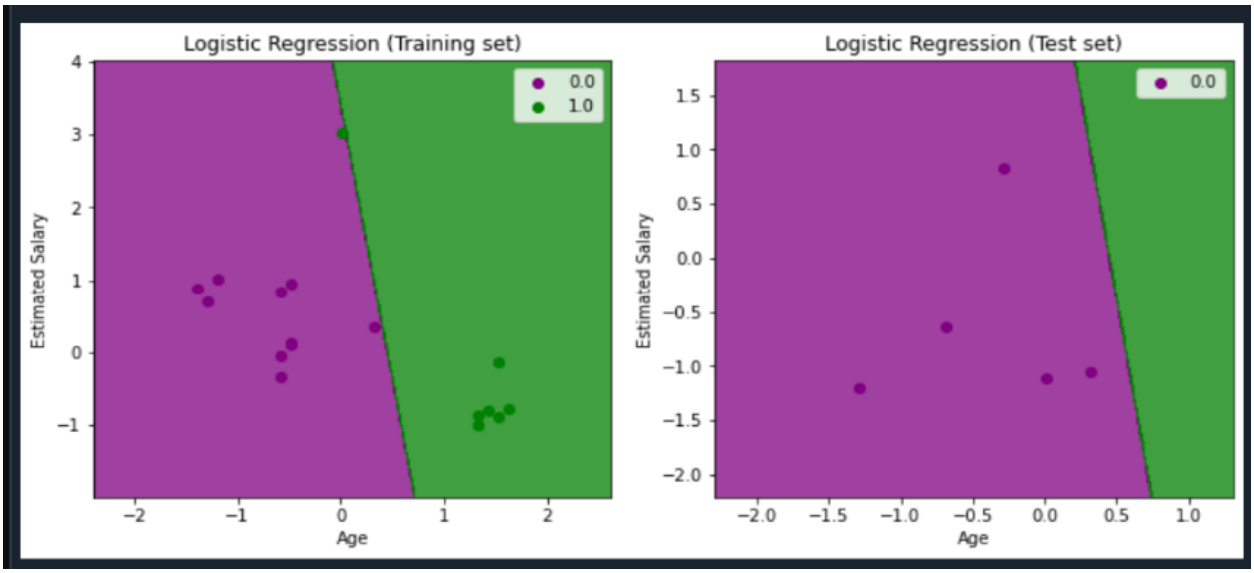
Confusion Matrix for SVM (Test set):

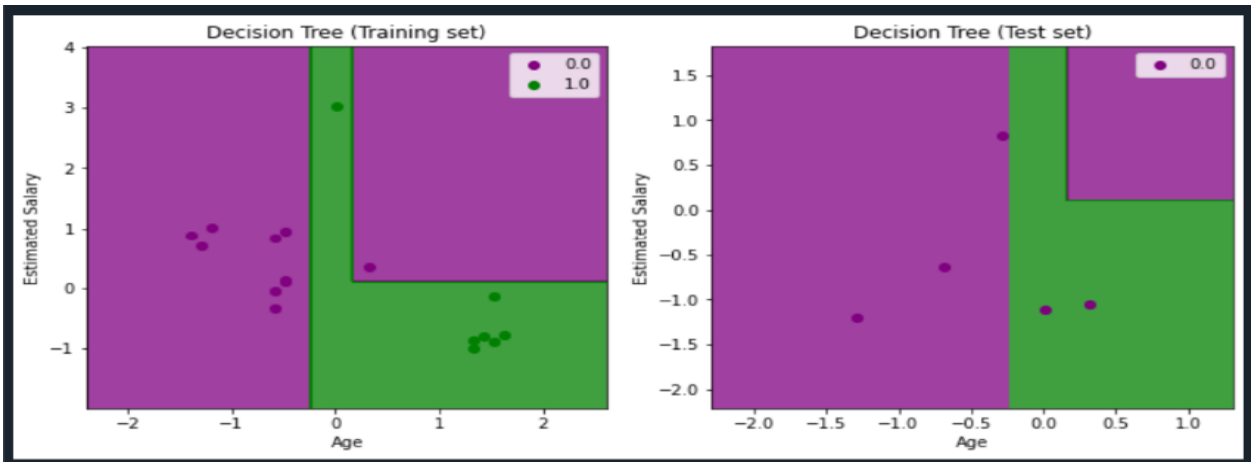
[[5]]

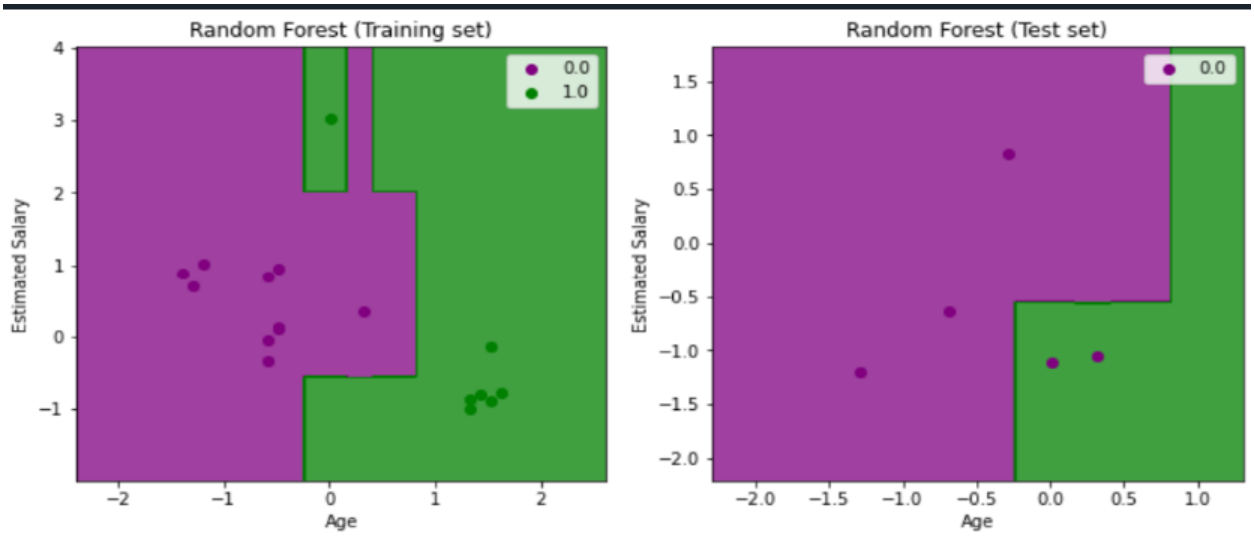
Confusion Matrix for kNN (Test set):

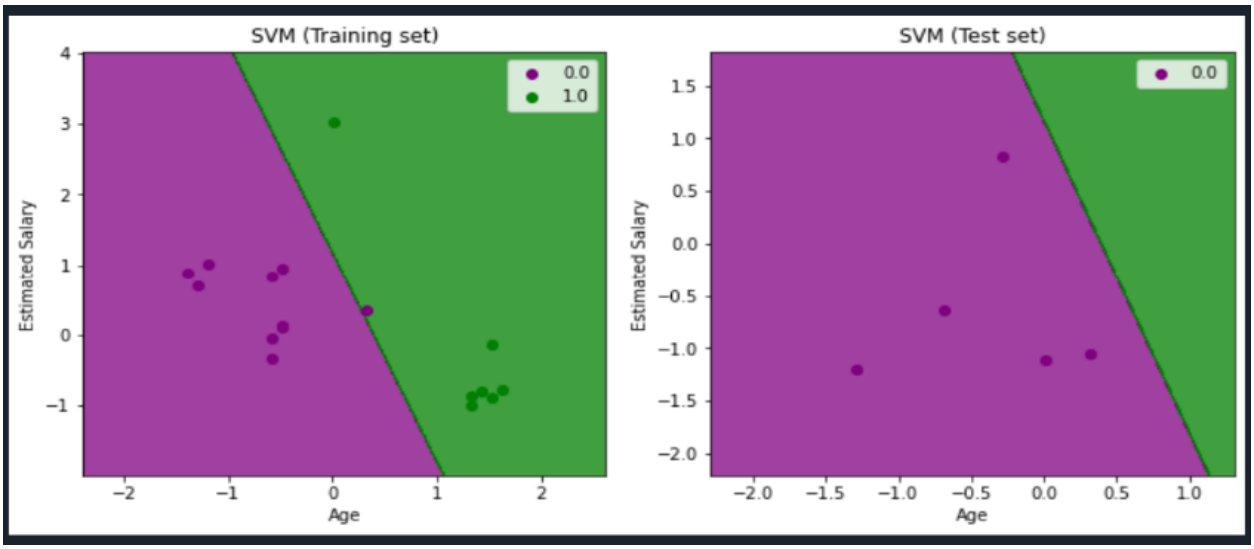
[[4 1]

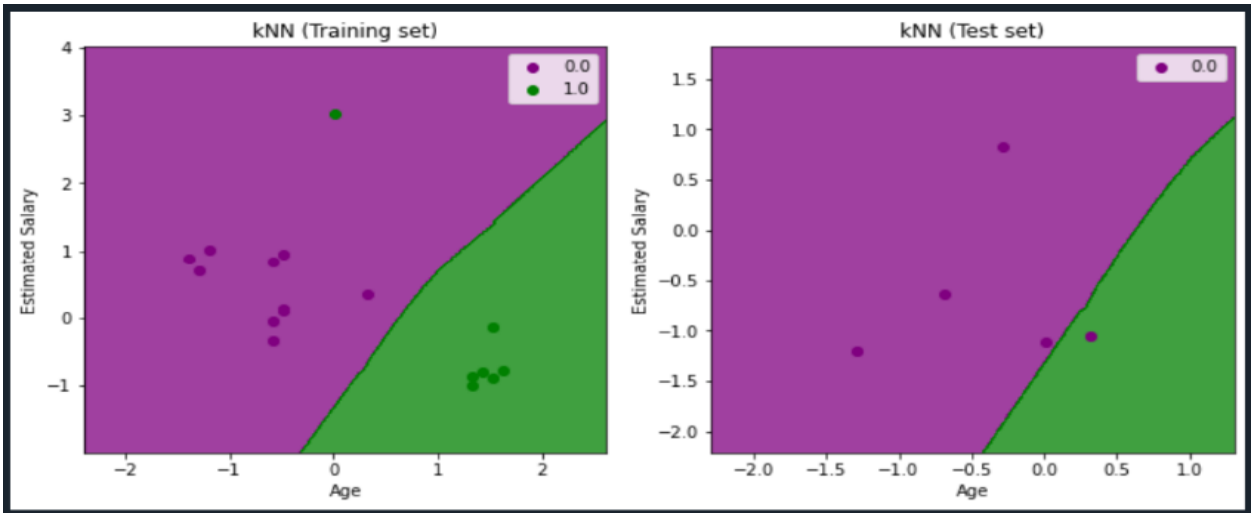
 [0 0]]


Logistic Regression (Training set)
Logistic Regression (Test set)
0.0
1.0
Estimated Salary
Age


Decision Tree (Training set)
Decision Tree (Test set)
0.0
1.0
Estimated Salary
Age


Random Forest (Training set)
Random Forest (Test set)
0.0
1.0
Estimated Salary
Age


SVM (Training set)
Estimated Salary
Age
0.0
1.0
SVM (Test set)
Estimated Salary
Age
0.0


kNN (Training set)
Estimated Salary
Age
0.0
1.0
kNN (Test set)
Estimated Salary
Age
0.0

Test Set Results:

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	1	1	1.0	1.00
Decision Tree	0.6	0.6	1.0	0.75
Random Forest	0.6	0.6	1.0	0.75
SVM	1	1	1.0	1.00
kNN	0.8	0.8	1.0	0.89

Conclusion: In conclusion, we have successfully trained and evaluated multiple classification models on the given dataset. Each model's performance metrics have been calculated and compared, allowing us to assess their effectiveness in predicting customer purchases based on age and estimated salary.