50)Create a Python program to read 'diabetes.csv' and evaluate the performance of a multivariate logistic regression model for multiclass classification. Calculate and display accuracy

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Read the CSV file
data = pd.read_csv('diabetes.csv')

# Split data into features (X) and target (y)
X = data.drop('target', axis=1)
y = data['target']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Initialize and train the logistic regression model
model = LogisticRegression(multi_class='multinomial',
solver='lbfgs', max_iter=1000)
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

51) Create a Python program to read 'diabetes.csv' and evaluate the performance of a multivariate logistic regression model for multiclass classification. Calculate and display recall.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import recall_score

# Read the CSV file
data = pd.read_csv('diabetes.csv')

# Split data into features (X) and target (y)
X = data.drop('target', axis=1)
y = data['target']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the logistic regression model
model = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=1000)
model.fit(X_train, y_train)
```

```
# Make predictions on the test set
y_pred = model.predict(X_test)

# Calculate recall
recall = recall_score(y_test, y_pred, average='weighted')
print("Recall:", recall)
```

52) Create a Python program to read 'HeartDisease.csv' and evaluate the performance of a multivariate logistic regression model for multiclass classification. Calculate and display metrics such as precision and F1-score

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import precision_score, f1_score

# Read the CSV file
data = pd.read_csv('HeartDisease.csv')

# Split data into features (X) and target (y)
X = data.drop('target', axis=1)
```

```
y = data['target']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Initialize and train the logistic regression model
model = LogisticRegression(multi_class='multinomial',
solver='lbfgs', max_iter=1000)
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Calculate precision
precision = precision_score(y_test, y_pred,
average='weighted')

# Calculate F1-score
f1 = f1_score(y_test, y_pred, average='weighted')

print("Precision:", precision)
print("F1-score:", f1)
```

53) Create a Python program that reads the contents of 'data.csv', Display metadata of the dataset and print the number of occurence of <'data_name'> in column <'col_name'>

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import precision_score, f1_score


# Read the CSV file

data = pd.read_csv('HeartDisease.csv')


# Split data into features (X) and target (y)

X = data.drop('target', axis=1)

y = data['target']


# Split data into train and test sets

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


# Initialize and train the logistic regression model

model = LogisticRegression(multi_class='multinomial',
solver='lbfgs', max_iter=1000)
```

```
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Calculate precision
precision = precision_score(y_test, y_pred,
average='weighted')

# Calculate F1-score
f1 = f1_score(y_test, y_pred, average='weighted')

print("Precision:", precision)
print("F1-score:", f1)
```

54) Create a Python program that reads the contents of 'data.csv', Display metadata of the dataset and replace NaN values with mean in the first column, median in the second column and mode in the third column

```
import pandas as pd

# Read the CSV file
data = pd.read_csv('data.csv')
```

```
# Display metadata of the dataset
print("Metadata of the dataset:")
print(data.info())

# Replace NaN values with mean, median, and mode in the first, second, and third columns respectively
for column in data.columns:
    if data[column].dtype != 'object':
        mean_value = data[column].mean()
        median_value = data[column].median()
        mode_value = data[column].mode()[0]

        data[column].fillna(mean_value, inplace=True)
        data[column].fillna(median_value, inplace=True)
        data[column].fillna(mode_value, inplace=True)

# Display the modified dataset
print("\nModified dataset with NaN values replaced:")
print(data.head())
```

55) write a python program to implement a K-Means clustering on 'dataset.csv' dataset and create a new column in the csv file, whose values corresponds to cluster

```
import pandas as pd
from sklearn.cluster import KMeans

# Read the CSV file
data = pd.read_csv('dataset.csv')

# Extract features
X = data.drop(columns=['cluster'])

# Perform K-Means clustering
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)

# Add cluster labels to the dataset
data['cluster'] = kmeans.labels_

# Write the modified dataset back to CSV file
data.to_csv('dataset_with_clusters.csv', index=False)

print("Clustering complete. Dataset with cluster labels saved as 'dataset_with_clusters.csv'.")
```

56) Write a python program to read a dataset and using Multivariate Logistic regression and Naive Bayes classifier, compare their accuracies

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

# Read the dataset
```

```
data = pd.read_csv('dataset.csv')

# Split data into features (X) and target (y)
X = data.drop(columns=['target'])
y = data['target']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Initialize and train Multivariate Logistic Regression model
logistic_reg_model = LogisticRegression(max_iter=1000)
logistic_reg_model.fit(X_train, y_train)

# Make predictions using logistic regression model
y_pred_logistic_reg = logistic_reg_model.predict(X_test)

# Calculate accuracy for logistic regression model
accuracy_logistic_reg = accuracy_score(y_test,
y_pred_logistic_reg)

# Initialize and train Naive Bayes classifier (GaussianNB)
naive_bayes_model = GaussianNB()
naive_bayes_model.fit(X_train, y_train)

# Make predictions using Naive Bayes classifier
y_pred_naive_bayes = naive_bayes_model.predict(X_test)

# Calculate accuracy for Naive Bayes classifier
accuracy_naive_bayes = accuracy_score(y_test,
y_pred_naive_bayes)

# Compare accuracies
```

```
print("Accuracy of Multivariate Logistic Regression:",
accuracy_logistic_reg)
print("Accuracy of Naive Bayes Classifier:",
accuracy_naive_bayes)
```

57) Write a python program to read a dataset and using Multivariate Logistic regression and Naive Bayes classifier, compare their precision

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import precision_score

# Read the dataset
data = pd.read_csv('dataset.csv')

# Split data into features (X) and target (y)
X = data.drop(columns=['target'])
y = data['target']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Initialize and train Multivariate Logistic Regression model
logistic_reg_model = LogisticRegression(max_iter=1000)
logistic_reg_model.fit(X_train, y_train)

# Make predictions using logistic regression model
y_pred_logistic_reg = logistic_reg_model.predict(X_test)
```

```
# Calculate precision for logistic regression model
precision_logistic_reg = precision_score(y_test,
y_pred_logistic_reg, average='weighted')

# Initialize and train Naive Bayes classifier (GaussianNB)
naive_bayes_model = GaussianNB()
naive_bayes_model.fit(X_train, y_train)

# Make predictions using Naive Bayes classifier
y_pred_naive_bayes = naive_bayes_model.predict(X_test)

# Calculate precision for Naive Bayes classifier
precision_naive_bayes = precision_score(y_test,
y_pred_naive_bayes, average='weighted')

# Compare precisions
print("Precision of Multivariate Logistic Regression:",
precision_logistic_reg)
print("Precision of Naive Bayes Classifier:",
precision_naive_bayes)
```

58) Display the metadata of a csv file

```
import pandas as pd

# Read the CSV file
data = pd.read_csv('your_file.csv')

# Display metadata of the dataset
print("Metadata of the dataset:")
print(data.info())
```

59) Write a python program to read a dataset and using Linaer Regression and multiple linear regression, compare their Mean Squared Error

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Read the dataset
data = pd.read_csv('dataset.csv')

# Assume 'target' is the target variable, and other columns are features
X = data.drop(columns=['target'])
y = data['target']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Linear Regression model
linear_reg_model = LinearRegression()
```

```
linear_reg_model.fit(X_train, y_train)

# Make predictions using Linear Regression model
y_pred_linear_reg = linear_reg_model.predict(X_test)

# Calculate Mean Squared Error for Linear Regression model
mse_linear_reg = mean_squared_error(y_test, y_pred_linear_reg)

# Train Multiple Linear Regression model
multiple_linear_reg_model = LinearRegression()
multiple_linear_reg_model.fit(X_train, y_train)

# Make predictions using Multiple Linear Regression model
y_pred_multiple_linear_reg = multiple_linear_reg_model.predict(X_test)

# Calculate Mean Squared Error for Multiple Linear Regression model
mse_multiple_linear_reg = mean_squared_error(y_test, y_pred_multiple_linear_reg)

# Compare Mean Squared Errors
```

```
print("Mean Squared Error (Linear Regression):", mse_linear_reg)
print("Mean Squared Error (Multiple Linear Regression):", mse_multiple_linear_reg)
```

60) Write a python program to read a dataset and using Multivariate Logistic regression and Naive Bayes classifier, compare their f1 score

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import f1_score

# Read the dataset
data = pd.read_csv('dataset.csv')

# Split data into features (X) and target (y)
X = data.drop(columns=['target'])
y = data['target']

# Split data into train and test sets
```

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

# Initialize and train Multivariate Logistic Regression model
logistic_reg_model = LogisticRegression(max_iter=1000)
logistic_reg_model.fit(X_train, y_train)

# Make predictions using logistic regression model
y_pred_logistic_reg = logistic_reg_model.predict(X_test)

# Calculate F1 score for logistic regression model
f1_logistic_reg = f1_score(y_test, y_pred_logistic_reg,
average='weighted')

# Initialize and train Naive Bayes classifier (GaussianNB)
naive_bayes_model = GaussianNB()
naive_bayes_model.fit(X_train, y_train)

# Make predictions using Naive Bayes classifier
y_pred_naive_bayes = naive_bayes_model.predict(X_test)

# Calculate F1 score for Naive Bayes classifier
f1_naive_bayes = f1_score(y_test, y_pred_naive_bayes,
average='weighted')
```

```
# Compare F1 scores
print("F1 Score of Multivariate Logistic Regression:", f1_logistic_reg)
print("F1 Score of Naive Bayes Classifier:", f1_naive_bayes)
```

61) Write a python program to read a dataset using Naive bayes classifier and logistic regression and compare their accuracies

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Read the dataset
data = pd.read_csv('dataset.csv')

# Split data into features (X) and target (y)
X = data.drop(columns=['target'])
y = data['target']

# Split data into train and test sets
```

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

# Initialize and train Naive Bayes classifier
naive_bayes_model = GaussianNB()
naive_bayes_model.fit(X_train, y_train)

# Make predictions using Naive Bayes classifier
y_pred_naive_bayes = naive_bayes_model.predict(X_test)

# Calculate accuracy for Naive Bayes classifier
accuracy_naive_bayes = accuracy_score(y_test, y_pred_naive_bayes)

# Initialize and train Logistic Regression model
logistic_regression_model = LogisticRegression(max_iter=1000)
logistic_regression_model.fit(X_train, y_train)

# Make predictions using Logistic Regression model
y_pred_logistic_regression = logistic_regression_model.predict(X_test)

# Calculate accuracy for Logistic Regression model
```

```
accuracy_logistic_regression = accuracy_score(y_test, y_pred_logistic_regression)

# Compare accuracies
print("Accuracy of Naive Bayes Classifier:", accuracy_naive_bayes)
print("Accuracy of Logistic Regression:", accuracy_logistic_regression)
```

62) Write a python program to read any dataset using linear regression and logistic regression and compare their R squared error

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import r2_score

# Read the dataset
data = pd.read_csv('dataset.csv')

# Split data into features (X) and target (y)
X = data.drop(columns=['target'])
```

```
y = data['target']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Initialize and train Linear Regression model
linear_regression_model = LinearRegression()
linear_regression_model.fit(X_train, y_train)

# Make predictions using Linear Regression model
y_pred_linear_regression =
linear_regression_model.predict(X_test)

# Calculate R-squared error for Linear Regression model
r_squared_error_linear_regression = r2_score(y_test,
y_pred_linear_regression)

# Print R-squared error for Linear Regression model
print("R-squared error of Linear Regression:",
r_squared_error_linear_regression)
```

63) write a python program to implement knn classifier on diabetes.csv and split model with test size=0.3 and keep number of neighbours to 5 and print classification report

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report

# Read the dataset
data = pd.read_csv('diabetes.csv')

# Split data into features (X) and target (y)
X = data.drop(columns=['Outcome'])
y = data['Outcome']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# Initialize and train KNN classifier
knn_classifier = KNeighborsClassifier(n_neighbors=5)
knn_classifier.fit(X_train, y_train)
```

```
# Make predictions using KNN classifier
y_pred = knn_classifier.predict(X_test)

# Print classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

64) write a python program to implement knn classifer on heart.csv ,split model with train size=0.8 and keep number of neighbours to 7 and print confusion matrix

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix

# Read the dataset
data = pd.read_csv('heart.csv')

# Split data into features (X) and target (y)
X = data.drop(columns=['target'])
y = data['target']

# Split data into train and test sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, random_state=42)

# Initialize and train KNN classifier
knn_classifier = KNeighborsClassifier(n_neighbors=7)
knn_classifier.fit(X_train, y_train)

# Make predictions using KNN classifier
y_pred = knn_classifier.predict(X_test)

# Print confusion matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

65) write a python program to implement adaboost classifier on HeartDisease.csv and print accuracy score

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score
```

```
# Read the dataset
data = pd.read_csv('HeartDisease.csv')

# Split data into features (X) and target (y)
X = data.drop(columns=['target'])
y = data['target']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Initialize and train AdaBoost classifier
adaboost_classifier = AdaBoostClassifier(n_estimators=50,
random_state=42)
adaboost_classifier.fit(X_train, y_train)

# Make predictions using AdaBoost classifier
y_pred = adaboost_classifier.predict(X_test)

# Calculate accuracy score
accuracy = accuracy_score(y_test, y_pred)

# Print accuracy score
print("Accuracy Score:", accuracy)
```

66) write a python program to implement support vector classifier on social.csv dataset and keep train size as 0.7 . Calculate and display precision score(hint: remove column UserID)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import precision_score
from sklearn.preprocessing import LabelEncoder

# Read the dataset
data = pd.read_csv('social.csv')

# Remove the 'UserID' column
data.drop(columns=['UserID'], inplace=True)

# Encode categorical variables
label_encoder = LabelEncoder()
data['Gender'] = label_encoder.fit_transform(data['Gender'])

# Split data into features (X) and target (y)
```

```
X = data.drop(columns=['Purchased'])
y = data['Purchased']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, random_state=42)

# Initialize and train Support Vector Classifier
svc_classifier = SVC(kernel='linear')
svc_classifier.fit(X_train, y_train)

# Make predictions using Support Vector Classifier
y_pred = svc_classifier.predict(X_test)

# Calculate precision score
precision = precision_score(y_test, y_pred)

# Print precision score
print("Precision Score:", precision)
```

67) write a python program to implement support vector classifier on social.csv dataset and keep train size as 0.7 . Calculate and display accuracy score(hint: remove column UserID)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder

# Read the dataset
data = pd.read_csv('social.csv')

# Remove the 'UserID' column
data.drop(columns=['UserID'], inplace=True)

# Encode categorical variables
label_encoder = LabelEncoder()
data['Gender'] = label_encoder.fit_transform(data['Gender'])

# Split data into features (X) and target (y)
X = data.drop(columns=['Purchased'])
y = data['Purchased']

# Split data into train and test sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, random_state=42)

# Initialize and train Support Vector Classifier
svc_classifier = SVC(kernel='linear')
svc_classifier.fit(X_train, y_train)

# Make predictions using Support Vector Classifier
y_pred = svc_classifier.predict(X_test)

# Calculate accuracy score
accuracy = accuracy_score(y_test, y_pred)

# Print accuracy score
print("Accuracy Score:", accuracy)
```

68) write a python program to implement adaboost classifier on social.csv dataset,split model with test size=0.2,use base estimator as Logistic regression and display the confusion matrix

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix

# Read the dataset
data = pd.read_csv('social.csv')

# Split data into features (X) and target (y)
X = data.drop(columns=['Purchased', 'UserID'])
y = data['Purchased']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Initialize and train base estimator (Logistic Regression)
base_estimator = LogisticRegression(max_iter=1000)
base_estimator.fit(X_train, y_train)

# Initialize and train AdaBoost classifier with base estimator
adaboost_classifier =
AdaBoostClassifier(base_estimator=base_estimator,
n_estimators=50, random_state=42)
adaboost_classifier.fit(X_train, y_train)
```

```
# Make predictions using AdaBoost classifier
y_pred = adaboost_classifier.predict(X_test)

# Calculate confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Print confusion matrix
print("Confusion Matrix:")
print(conf_matrix)
```

69)write a python program to implement adaboost classifier on social.csv dataset,split model with test size=0.2,use base estimator as SVC and display the confusion matrix

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix

# Read the dataset
data = pd.read_csv('social.csv')

# Split data into features (X) and target (y)
```

```
X = data.drop(columns=['Purchased', 'UserID'])
y = data['Purchased']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Initialize and train base estimator (Support Vector
Classifier)
base_estimator = SVC(kernel='linear')
base_estimator.fit(X_train, y_train)

# Initialize and train AdaBoost classifier with base estimator
adaboost_classifier =
AdaBoostClassifier(base_estimator=base_estimator,
n_estimators=50, random_state=42)
adaboost_classifier.fit(X_train, y_train)

# Make predictions using AdaBoost classifier
y_pred = adaboost_classifier.predict(X_test)

# Calculate confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
```

```
# Print confusion matrix
print("Confusion Matrix:")
print(conf_matrix)
```

70)write a python program to implement adaboost classifier on social.csv dataset,split model with test size=0.3,use base estimator as SVC and display the confusion matrix

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix

# Read the dataset
data = pd.read_csv('social.csv')

# Split data into features (X) and target (y)
X = data.drop(columns=['Purchased', 'UserID'])
y = data['Purchased']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# Initialize and train base estimator (Support Vector
Classifier)
base_estimator = SVC(kernel='linear')
base_estimator.fit(X_train, y_train)

# Initialize and train AdaBoost classifier with base estimator
```

```
adaboost_classifier =
AdaBoostClassifier(base_estimator=base_estimator,
n_estimators=50, random_state=42)
adaboost_classifier.fit(X_train, y_train)

# Make predictions using AdaBoost classifier
y_pred = adaboost_classifier.predict(X_test)

# Calculate confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Print confusion matrix
print("Confusion Matrix:")
print(conf_matrix)
```

71) write a python program to implement adaboost classifier on social.csv dataset,split model with test size=0.2,use base estimator as SVC and display the accuracy score

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Read the dataset
data = pd.read_csv('social.csv')

# Split data into features (X) and target (y)
X = data.drop(columns=['Purchased', 'UserID'])
y = data['Purchased']

# Split data into train and test sets
```

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

# Initialize and train base estimator (Support Vector
Classifier)
base_estimator = SVC(kernel='linear')
base_estimator.fit(X_train, y_train)

# Initialize and train AdaBoost classifier with base estimator
adaboost_classifier =
AdaBoostClassifier(base_estimator=base_estimator,
n_estimators=50, random_state=42)
adaboost_classifier.fit(X_train, y_train)

# Make predictions using AdaBoost classifier
y_pred = adaboost_classifier.predict(X_test)

# Calculate accuracy score
accuracy = accuracy_score(y_test, y_pred)

# Print accuracy score
print("Accuracy Score:", accuracy)
```

72) write a python program to implement Decision tree classifier on iris.csv dataset,split model with train size=0.8, and print accuracy score(use labelencoder from sklearn to convert column Species from categorical to numeric)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
```

```
# Read the dataset
data = pd.read_csv('iris.csv')

# Use LabelEncoder to convert categorical column 'Species' to
numeric
label_encoder = LabelEncoder()
data['Species'] = label_encoder.fit_transform(data['Species'])

# Split data into features (X) and target (y)
X = data.drop(columns=['Species'])
y = data['Species']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
train_size=0.8, random_state=42)

# Initialize and train Decision Tree Classifier
decision_tree_classifier =
DecisionTreeClassifier(random_state=42)
decision_tree_classifier.fit(X_train, y_train)

# Make predictions using Decision Tree Classifier
y_pred = decision_tree_classifier.predict(X_test)

# Calculate accuracy score
accuracy = accuracy_score(y_test, y_pred)

# Print accuracy score
print("Accuracy Score:", accuracy)
```

73) write a python program to implement Decision tree classifier on iris.csv dataset,split model with train size=0.8,

and print precision score(use labelencoder from sklearn to convert column Species from categorical to numeric)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import precision_score
from sklearn.preprocessing import LabelEncoder

# Read the dataset
data = pd.read_csv('iris.csv')

# Use LabelEncoder to convert categorical column 'Species' to
numeric
label_encoder = LabelEncoder()
data['Species'] = label_encoder.fit_transform(data['Species'])

# Split data into features (X) and target (y)
X = data.drop(columns=['Species'])
y = data['Species']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
train_size=0.8, random_state=42)

# Initialize and train Decision Tree Classifier
decision_tree_classifier =
DecisionTreeClassifier(random_state=42)
decision_tree_classifier.fit(X_train, y_train)

# Make predictions using Decision Tree Classifier
y_pred = decision_tree_classifier.predict(X_test)
```

```
# Calculate precision score
precision = precision_score(y_test, y_pred,
average='weighted')

# Print precision score
print("Precision Score:", precision)
```

74) write a python program to implement Decision tree classifier on iris.csv dataset,split model with train size=0.8, and print accuracy score,f1 score(use labelencoder from sklearn to convert column Species from categorical to numeric)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, f1_score
from sklearn.preprocessing import LabelEncoder

# Read the dataset
data = pd.read_csv('iris.csv')

# Use LabelEncoder to convert categorical column 'Species' to
numeric
label_encoder = LabelEncoder()
data['Species'] = label_encoder.fit_transform(data['Species'])

# Split data into features (X) and target (y)
X = data.drop(columns=['Species'])
y = data['Species']

# Split data into train and test sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
train_size=0.8, random_state=42)

# Initialize and train Decision Tree Classifier
decision_tree_classifier =
DecisionTreeClassifier(random_state=42)
decision_tree_classifier.fit(X_train, y_train)

# Make predictions using Decision Tree Classifier
y_pred = decision_tree_classifier.predict(X_test)

# Calculate accuracy score
accuracy = accuracy_score(y_test, y_pred)

# Calculate F1 score
f1 = f1_score(y_test, y_pred, average='weighted')

# Print accuracy score and F1 score
print("Accuracy Score:", accuracy)
print("F1 Score:", f1)
```

75) write a python program to implement Decision tree classifier on iris.csv dataset,split model with train size=0.8, and print f1 score(use labelencoder from sklearn to convert column Species from categorical to numeric)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import f1_score
from sklearn.preprocessing import LabelEncoder

# Read the dataset
```

```
data = pd.read_csv('iris.csv')

# Use LabelEncoder to convert categorical column 'Species' to
numeric
label_encoder = LabelEncoder()
data['Species'] = label_encoder.fit_transform(data['Species'])

# Split data into features (X) and target (y)
X = data.drop(columns=['Species'])
y = data['Species']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
train_size=0.8, random_state=42)

# Initialize and train Decision Tree Classifier
decision_tree_classifier =
DecisionTreeClassifier(random_state=42)
decision_tree_classifier.fit(X_train, y_train)

# Make predictions using Decision Tree Classifier
y_pred = decision_tree_classifier.predict(X_test)

# Calculate F1 score
f1 = f1_score(y_test, y_pred, average='weighted')

# Print F1 score
print("F1 Score:", f1)
```

76) write a python program to implement a classification algorithm on cell_samples.csv and display accuracy (hint:remove the column ID)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Read the dataset
data = pd.read_csv('cell_samples.csv')

# Remove the 'ID' column
data.drop(columns=['ID'], inplace=True)

# Split data into features (X) and target (y)
X = data.drop(columns=['Class'])
y = data['Class']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Initialize and train Random Forest Classifier
random_forest_classifier =
RandomForestClassifier(random_state=42)
random_forest_classifier.fit(X_train, y_train)

# Make predictions using Random Forest Classifier
y_pred = random_forest_classifier.predict(X_test)

# Calculate accuracy score
accuracy = accuracy_score(y_test, y_pred)

# Print accuracy score
print("Accuracy Score:", accuracy)
```

77) write a python program to implement a classification algorithm on cell_samples.csv and display classification report (hint:remove the column ID

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

# Read the dataset
data = pd.read_csv('cell_samples.csv')

# Remove the 'ID' column
data.drop(columns=['ID'], inplace=True)

# Split data into features (X) and target (y)
X = data.drop(columns=['Class'])
y = data['Class']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train Random Forest Classifier
```

```
random_forest_classifier = RandomForestClassifier(random_state=42)
random_forest_classifier.fit(X_train, y_train)

# Make predictions using Random Forest Classifier
y_pred = random_forest_classifier.predict(X_test)

# Generate classification report
report = classification_report(y_test, y_pred)

# Print classification report
print("Classification Report:\n", report)
```

78) write a python program to implement a classification algorithm on cell_samples.csv and display confusion matrix (hint:remove the column ID)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix

# Read the dataset
data = pd.read_csv('cell_samples.csv')
```

```
# Remove the 'ID' column
data.drop(columns=['ID'], inplace=True)

# Split data into features (X) and target (y)
X = data.drop(columns=['Class'])
y = data['Class']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Initialize and train Random Forest Classifier
random_forest_classifier =
RandomForestClassifier(random_state=42)
random_forest_classifier.fit(X_train, y_train)

# Make predictions using Random Forest Classifier
y_pred = random_forest_classifier.predict(X_test)

# Generate confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Print confusion matrix
```

```
print("Confusion Matrix:")
print(conf_matrix)
```

79) write a python program to implement Support vector classifier on cell_samples.csv and display confusion matrix(hint:remove the column ID)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import LabelEncoder

# Read the dataset
data = pd.read_csv('cell_samples.csv')

# Remove the 'ID' column
data.drop(columns=['ID'], inplace=True)

# Encode the 'Class' column to numerical values
label_encoder = LabelEncoder()
data['Class'] = label_encoder.fit_transform(data['Class'])

# Split data into features (X) and target (y)
```

```
X = data.drop(columns=['Class'])
y = data['Class']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Initialize and train Support Vector Classifier
svc_classifier = SVC(kernel='linear')
svc_classifier.fit(X_train, y_train)

# Make predictions using Support Vector Classifier
y_pred = svc_classifier.predict(X_test)

# Generate confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Print confusion matrix
print("Confusion Matrix:")
print(conf_matrix)
```

80) write a python program to implement support vector classifier on cell_samples.csv and display classification report(hint:remove the column ID)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelEncoder

# Read the dataset
data = pd.read_csv('cell_samples.csv')

# Remove the 'ID' column
data.drop(columns=['ID'], inplace=True)

# Encode the 'Class' column to numerical values
label_encoder = LabelEncoder()
data['Class'] = label_encoder.fit_transform(data['Class'])

# Split data into features (X) and target (y)
X = data.drop(columns=['Class'])
y = data['Class']

# Split data into train and test sets
```

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

# Initialize and train Support Vector Classifier
svc_classifier = SVC(kernel='linear')
svc_classifier.fit(X_train, y_train)

# Make predictions using Support Vector Classifier
y_pred = svc_classifier.predict(X_test)

# Generate classification report
report = classification_report(y_test, y_pred)

# Print classification report
print("Classification Report:\n", report)
```

81) write a python program to implement support vector classifier on cell_samples.csv and display accuracy(hint:remove the column ID)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder

# Read the dataset
data = pd.read_csv('cell_samples.csv')

# Remove the 'ID' column
data.drop(columns=['ID'], inplace=True)

# Encode the 'Class' column to numerical values
label_encoder = LabelEncoder()
data['Class'] = label_encoder.fit_transform(data['Class'])

# Split data into features (X) and target (y)
X = data.drop(columns=['Class'])
y = data['Class']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Initialize and train Support Vector Classifier
svc_classifier = SVC(kernel='linear')
svc_classifier.fit(X_train, y_train)
```

```
# Make predictions using Support Vector Classifier
y_pred = svc_classifier.predict(X_test)

# Calculate accuracy score
accuracy = accuracy_score(y_test, y_pred)

# Print accuracy score
print("Accuracy Score:", accuracy)
```

82) write a python program to implement Decision Tree classifier on cell_samples.csv and display accuracy(hint:remove the column ID)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# Read the dataset
data = pd.read_csv('cell_samples.csv')

# Remove the 'ID' column
data.drop(columns=['ID'], inplace=True)
```

```
# Split data into features (X) and target (y)
X = data.drop(columns=['Class'])
y = data['Class']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Initialize and train Decision Tree Classifier
decision_tree_classifier =
DecisionTreeClassifier(random_state=42)
decision_tree_classifier.fit(X_train, y_train)

# Make predictions using Decision Tree Classifier
y_pred = decision_tree_classifier.predict(X_test)

# Calculate accuracy score
accuracy = accuracy_score(y_test, y_pred)

# Print accuracy score
print("Accuracy Score:", accuracy)
```

83) write a python program to implement Decision Tree classifier on cell_samples.csv and display classification report(hint:remove the column ID)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report

# Read the dataset
data = pd.read_csv('cell_samples.csv')

# Remove the 'ID' column
data.drop(columns=['ID'], inplace=True)

# Split data into features (X) and target (y)
X = data.drop(columns=['Class'])
y = data['Class']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train Decision Tree Classifier
```

```
decision_tree_classifier = DecisionTreeClassifier(random_state=42)
decision_tree_classifier.fit(X_train, y_train)

# Make predictions using Decision Tree Classifier
y_pred = decision_tree_classifier.predict(X_test)

# Generate classification report
report = classification_report(y_test, y_pred)

# Print classification report
print("Classification Report:\n", report)
```

84) write a python program to implement Decision Tree Classifier on cell_samples.csv and display confusion matrix(hint:remove the column ID)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix

# Read the dataset
data = pd.read_csv('cell_samples.csv')
```

```
# Remove the 'ID' column
data.drop(columns=['ID'], inplace=True)

# Split data into features (X) and target (y)
X = data.drop(columns=['Class'])
y = data['Class']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Initialize and train Decision Tree Classifier
decision_tree_classifier =
DecisionTreeClassifier(random_state=42)
decision_tree_classifier.fit(X_train, y_train)

# Make predictions using Decision Tree Classifier
y_pred = decision_tree_classifier.predict(X_test)

# Generate confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Print confusion matrix
```

```
print("Confusion Matrix:")
print(conf_matrix)
```

85) write a python program to implement knn classifier on cell_samples.csv and display accuracy(hint:remove the column ID)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder

# Read the dataset
data = pd.read_csv('cell_samples.csv')

# Remove the 'ID' column
data.drop(columns=['ID'], inplace=True)

# Encode the 'Class' column to numerical values
label_encoder = LabelEncoder()
data['Class'] = label_encoder.fit_transform(data['Class'])

# Split data into features (X) and target (y)
```

```
X = data.drop(columns=['Class'])
y = data['Class']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train KNN Classifier
knn_classifier = KNeighborsClassifier()
knn_classifier.fit(X_train, y_train)

# Make predictions using KNN Classifier
y_pred = knn_classifier.predict(X_test)

# Calculate accuracy score
accuracy = accuracy_score(y_test, y_pred)

# Print accuracy score
print("Accuracy Score:", accuracy)
```

86) write a python program to implement knn classifier on cell_samples.csv and display confusion matrix(hint:remove the column ID)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import LabelEncoder

# Read the dataset
data = pd.read_csv('cell_samples.csv')

# Remove the 'ID' column
data.drop(columns=['ID'], inplace=True)

# Encode the 'Class' column to numerical values
label_encoder = LabelEncoder()
data['Class'] = label_encoder.fit_transform(data['Class'])

# Split data into features (X) and target (y)
X = data.drop(columns=['Class'])
y = data['Class']

# Split data into train and test sets
```

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

# Initialize and train KNN Classifier
knn_classifier = KNeighborsClassifier()
knn_classifier.fit(X_train, y_train)

# Make predictions using KNN Classifier
y_pred = knn_classifier.predict(X_test)

# Generate confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Print confusion matrix
print("Confusion Matrix:")
print(conf_matrix)
```

87) write a python program to implement knn classifier on cell_samples.csv and display classification report(hint:remove the column ID)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelEncoder

# Read the dataset
data = pd.read_csv('cell_samples.csv')

# Remove the 'ID' column
data.drop(columns=['ID'], inplace=True)

# Encode the 'Class' column to numerical values
label_encoder = LabelEncoder()
data['Class'] = label_encoder.fit_transform(data['Class'])

# Split data into features (X) and target (y)
X = data.drop(columns=['Class'])
y = data['Class']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Initialize and train KNN Classifier
knn_classifier = KNeighborsClassifier()
```

```
knn_classifier.fit(X_train, y_train)

# Make predictions using KNN Classifier
y_pred = knn_classifier.predict(X_test)

# Generate classification report
report = classification_report(y_test, y_pred)

# Print classification report
print("Classification Report:\n", report)
```

88) write a python program to implement regression algorithm on Salary data.csv and print mean absolute error(hint:remove the column gender,use labelencoder from sklearn to convert columns Education Level,Job Title or use pd.get_dummies function and remove the null values by giving pd.dropna())

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
from sklearn.preprocessing import LabelEncoder

# Read the dataset
data = pd.read_csv('Salary data.csv')
```

```
# Remove the 'Gender' column
data.drop(columns=['Gender'], inplace=True)

# Encode the 'Education Level' and 'Job Title' columns using LabelEncoder
label_encoder = LabelEncoder()
data['Education Level'] = label_encoder.fit_transform(data['Education Level'])
data['Job Title'] = label_encoder.fit_transform(data['Job Title'])

# Remove null values
data.dropna(inplace=True)

# Split data into features (X) and target (y)
X = data.drop(columns=['Salary'])
y = data['Salary']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train Linear Regression model
```

```
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Calculate mean absolute error
mae = mean_absolute_error(y_test, y_pred)

# Print mean absolute error
print("Mean Absolute Error:", mae)
```

89) write a python program to implement regression algorithm on Salary data.csv and print mean Squared error(hint: remove the column gender,use labelencoder from sklearn to convert columns Education level,Job title or use pd.get_dummies function and remove the null values by using pd.dropna()

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import LabelEncoder

# Read the dataset
```

```
data = pd.read_csv('Salary data.csv')

# Remove the 'Gender' column
data.drop(columns=['Gender'], inplace=True)

# Encode the 'Education Level' and 'Job Title' columns using LabelEncoder
label_encoder = LabelEncoder()
data['Education Level'] = label_encoder.fit_transform(data['Education Level'])
data['Job Title'] = label_encoder.fit_transform(data['Job Title'])

# Remove null values
data.dropna(inplace=True)

# Split data into features (X) and target (y)
X = data.drop(columns=['Salary'])
y = data['Salary']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Initialize and train Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Calculate mean squared error
mse = mean_squared_error(y_test, y_pred)

# Print mean squared error
print("Mean Squared Error:", mse)
```

90) write a python program to implement simple linear regression algorithm on Salary data.csv and print mean Squared error(hint: remove the columns gender,Education level,Job title and remove the null values by using pd.dropna())

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Read the dataset
```

```
data = pd.read_csv('Salary data.csv')

# Remove the 'Gender', 'Education Level', and 'Job Title' columns
data.drop(columns=['Gender', 'Education Level', 'Job Title'], inplace=True)

# Remove null values
data.dropna(inplace=True)

# Split data into features (X) and target (y)
X = data.drop(columns=['Salary'])
y = data['Salary']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)
```

```
# Calculate mean squared error
mse = mean_squared_error(y_test, y_pred)

# Print mean squared error
print("Mean Squared Error:", mse)
```

91) write a python program to implement linear regression on Salary data.csv and display mean absolute error(hint:remove the column gender,use labelencoder from sklearn to convert columns Education level,Job title(categorical to numeric) or use pd.get_dummies function and remove the null values by using pd.dropna function)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
from sklearn.preprocessing import LabelEncoder

# Read the dataset
data = pd.read_csv('Salary data.csv')

# Remove the 'Gender' column
```

```
data.drop(columns=['Gender'], inplace=True)

# Encode the 'Education Level' and 'Job Title' columns using LabelEncoder
label_encoder = LabelEncoder()
data['Education Level'] = label_encoder.fit_transform(data['Education Level'])
data['Job Title'] = label_encoder.fit_transform(data['Job Title'])

# Remove null values
data.dropna(inplace=True)

# Split data into features (X) and target (y)
X = data.drop(columns=['Salary'])
y = data['Salary']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)
```

```
# Make predictions
y_pred = model.predict(X_test)

# Calculate mean absolute error
mae = mean_absolute_error(y_test, y_pred)

# Print mean absolute error
print("Mean Absolute Error:", mae)
```

92) write a python program to implement linear regression on Salary data.csv and display mean squared error(hint:remove the column gender,use labelencoder from sklearn to convert columns Education level,Job title(categorical to numeric) or use pd.get_dummies function and remove the null values by using pd.dropna function)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import LabelEncoder

# Read the dataset
data = pd.read_csv('Salary data.csv')
```

```
# Remove the 'Gender' column
data.drop(columns=['Gender'], inplace=True)

# Encode the 'Education Level' and 'Job Title' columns using LabelEncoder
label_encoder = LabelEncoder()
data['Education Level'] = label_encoder.fit_transform(data['Education Level'])
data['Job Title'] = label_encoder.fit_transform(data['Job Title'])

# Remove null values
data.dropna(inplace=True)

# Split data into features (X) and target (y)
X = data.drop(columns=['Salary'])
y = data['Salary']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train Linear Regression model
```

```
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Calculate mean squared error
mse = mean_squared_error(y_test, y_pred)

# Print mean squared error
print("Mean Squared Error:", mse)
```

93) write a python to implement linear regression on Salary data.csv and display Residual(hint:remove the column gender,use labelencoder from sklearn to convert columns Education level,Job Title (categorical to numeric) or use pd.get_dummies function and remove the null values by using pd.dropna function)

```
 import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
```

```
# Read the dataset
data = pd.read_csv('Salary data.csv')

# Remove the 'Gender' column
data.drop(columns=['Gender'], inplace=True)

# Encode the 'Education Level' and 'Job Title' columns using LabelEncoder
label_encoder = LabelEncoder()
data['Education Level'] = label_encoder.fit_transform(data['Education Level'])
data['Job Title'] = label_encoder.fit_transform(data['Job Title'])

# Remove null values
data.dropna(inplace=True)

# Split data into features (X) and target (y)
X = data.drop(columns=['Salary'])
y = data['Salary']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Initialize and train Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Calculate residuals
residuals = y_test - y_pred

# Plot residuals
plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_pred, y=residuals)
plt.axhline(y=0, color='red', linestyle='--')
plt.xlabel('Predicted Salary')
plt.ylabel('Residual')
plt.title('Residual Plot')
plt.show()
```

94) write a python to implement linear regression on Salary data.csv and display R squared error(hint:remove the column gender,use labelencoder from sklearn to convert columns Education level,Job Title (categorical to numeric) or use pd.get_dummies function and remove the null values by using pd.dropna function)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.preprocessing import LabelEncoder

# Read the dataset
data = pd.read_csv('Salary data.csv')

# Remove the 'Gender' column
data.drop(columns=['Gender'], inplace=True)

# Encode the 'Education Level' and 'Job Title' columns using LabelEncoder
label_encoder = LabelEncoder()
data['Education Level'] = label_encoder.fit_transform(data['Education Level'])
data['Job Title'] = label_encoder.fit_transform(data['Job Title'])

# Remove null values
```

```
data.dropna(inplace=True)

# Split data into features (X) and target (y)
X = data.drop(columns=['Salary'])
y = data['Salary']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Initialize and train Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Calculate R-squared error
r2_error = r2_score(y_test, y_pred)

# Print R-squared error
print("R-squared Error:", r2_error)
```

95) write a python program to decision tree regession on Salary data.csv and display R squared error(hint:remove the column gender,use labelencoder from sklearn to convert columns Education level,Job Title(categorical to numeric) or use pd.get_dummies function and remove the null values by using pd.dropna function)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score
from sklearn.preprocessing import LabelEncoder

# Read the dataset
data = pd.read_csv('Salary data.csv')

# Remove the 'Gender' column
data.drop(columns=['Gender'], inplace=True)

# Encode the 'Education Level' and 'Job Title' columns using LabelEncoder
label_encoder = LabelEncoder()
data['Education Level'] = label_encoder.fit_transform(data['Education Level'])
```

```
data['Job Title'] = label_encoder.fit_transform(data['Job Title'])

# Remove null values
data.dropna(inplace=True)

# Split data into features (X) and target (y)
X = data.drop(columns=['Salary'])
y = data['Salary']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train Decision Tree Regressor
model = DecisionTreeRegressor()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Calculate R-squared error
r2_error = r2_score(y_test, y_pred)
```

```
# Print R-squared error
print("R-squared Error:", r2_error)
```

96) write a python program to decision tree regession on Salary data.csv and display mean squared error(hint:remove the column gender,use labelencoder from sklearn to convert columns Education level,Job Title(categorical to numeric) or use pd.get_dummies function and remove the null values by using pd.dropna function)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import LabelEncoder

# Read the dataset
data = pd.read_csv('Salary data.csv')

# Remove the 'Gender' column
data.drop(columns=['Gender'], inplace=True)

# Encode the 'Education Level' and 'Job Title' columns using LabelEncoder
```

```
label_encoder = LabelEncoder()
data['Education Level'] = label_encoder.fit_transform(data['Education Level'])
data['Job Title'] = label_encoder.fit_transform(data['Job Title'])

# Remove null values
data.dropna(inplace=True)

# Split data into features (X) and target (y)
X = data.drop(columns=['Salary'])
y = data['Salary']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train Decision Tree Regressor
model = DecisionTreeRegressor()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)
```

```
# Calculate mean squared error
mse = mean_squared_error(y_test, y_pred)

# Print mean squared error
print("Mean Squared Error:", mse)
```

97) write a python program to decision tree regression on Salary data.csv and display mean absolute error(hint:remove the column gender,use labelencoder from sklearn to convert columns Education level,Job Title(categorical to numeric) or use pd.get_dummies function and remove the null values by using pd.dropna function)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.preprocessing import LabelEncoder

# Read the dataset
data = pd.read_csv('Salary data.csv')

# Remove the 'Gender' column
data.drop(columns=['Gender'], inplace=True)
```

```
# Encode the 'Education Level' and 'Job Title' columns using LabelEncoder
label_encoder = LabelEncoder()
data['Education Level'] = label_encoder.fit_transform(data['Education Level'])
data['Job Title'] = label_encoder.fit_transform(data['Job Title'])

# Remove null values
data.dropna(inplace=True)

# Split data into features (X) and target (y)
X = data.drop(columns=['Salary'])
y = data['Salary']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train Decision Tree Regressor
model = DecisionTreeRegressor()
model.fit(X_train, y_train)

# Make predictions
```

```
y_pred = model.predict(X_test)

# Calculate mean absolute error
mae = mean_absolute_error(y_test, y_pred)

# Print mean absolute error
print("Mean Absolute Error:", mae)
```

98) write a python program to decision tree regression on Salary data.csv and display Residual(hint:remove the column gender,use labelencoder from sklearn to convert columns Education level,Job Title(categorical to numeric) or use pd.get_dummies function and remove the null values by using pd.dropna function)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder

# Read the dataset
data = pd.read_csv('Salary data.csv')
```

```
# Remove the 'Gender' column
data.drop(columns=['Gender'], inplace=True)

# Encode the 'Education Level' and 'Job Title' columns using LabelEncoder
label_encoder = LabelEncoder()
data['Education Level'] = label_encoder.fit_transform(data['Education Level'])
data['Job Title'] = label_encoder.fit_transform(data['Job Title'])

# Remove null values
data.dropna(inplace=True)

# Split data into features (X) and target (y)
X = data.drop(columns=['Salary'])
y = data['Salary']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train Decision Tree Regressor
model = DecisionTreeRegressor()
```

```
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Calculate residuals
residuals = y_test - y_pred

# Plot residuals
plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_pred, y=residuals)
plt.axhline(y=0, color='red', linestyle='--')
plt.xlabel('Predicted Salary')
plt.ylabel('Residual')
plt.title('Residual Plot')
plt.show()
```

99) write a python program to implement simple linear regression algorithm on Salary data.csv and print R Squared error(hint: remove the columns gender,Education level,Job title and remove the null values by using pd.dropna())

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

# Read the dataset
data = pd.read_csv('Salary data.csv')

# Remove the 'Gender', 'Education level', and 'Job title' columns
data.drop(columns=['Gender', 'Education level', 'Job title'], inplace=True)

# Remove null values
data.dropna(inplace=True)

# Split data into features (X) and target (y)
X = data.drop(columns=['Salary'])
y = data['Salary']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train Linear Regression model
model = LinearRegression()
```

```
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Calculate R-squared error
r2_error = r2_score(y_test, y_pred)

# Print R-squared error
print("R-squared Error:", r2_error)
```

100) write a python program to implement simple linear regression algorithm on Salary data.csv and print Residual(hint:remove the columns gender,Education level,Job Title and remove the null avlues by using pd.dropna function

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import seaborn as sns

# Read the dataset
data = pd.read_csv('Salary data.csv')
```

```
# Remove the 'Gender', 'Education level', and 'Job Title' columns
data.drop(columns=['Gender', 'Education level', 'Job Title'], inplace=True)

# Remove null values
data.dropna(inplace=True)

# Split data into features (X) and target (y)
X = data.drop(columns=['Salary'])
y = data['Salary']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)
```

```
# Calculate residuals
residuals = y_test - y_pred

# Plot residuals
plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_pred, y=residuals)
plt.axhline(y=0, color='red', linestyle='--')
plt.xlabel('Predicted Salary')
plt.ylabel('Residual')
plt.title('Residual Plot')
plt.show()
```