Task 4

Suggest a suitable MBA specialization for the candidates based on their profile using "Mba_Admission" dataset.

□ Independent attributes: pre_score, Age_in_years, Percentage_in_10_Class, Percentage_in_12_Class, Percentage_in_Under_Graduate, post_score, Gender, STATE, Previous_Degree, Marital_status, Place_you_belong_to and perceived#Job#Skill

□ Dependent attribute: Specialization

□ Constraints:

o Split 80% data for training and 20% data for testing

o Use any classification techniques, correlation and label encoding / one hot encoding techniques

o Calculate the accuracy, precision, recall and F1 score for the final classification model#importing necessary modules

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
#Reading data
data = pd.read_csv("Mba_admission.csv")
# Split data into features (X) and target (y)
X = data.drop(columns=["s_no","Specialization"])
y = data["Specialization"]
# Encode categorical variables
le = LabelEncoder()
X_encoded = X.apply(le.fit_transform)
# Split data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)
# Initialize and train the model (e.g., Random Forest)
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
# Make predictions
y_pred = model.predict(X_test)
# Evaluate model performance
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average="weighted")
recall = recall_score(y_test, y_pred, average="weighted")
f1 = f1_score(y_test, y_pred, average="weighted")
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1-score: {f1:.2f}")
```

Task 2

Identify the chance for a movie getting the Oscar using “movie_classification“ dataset.

▪ Independent attributes: Marketing expense, Production expense, Multiplex coverage, Budget, Movie_length, Lead_ Actor_Rating, Lead_Actress_rating, Director_rating, Producer_rating, Critic_rating, Trailer_views, 3D_available, Time_taken, Twitter_hastags, Genre, Avg_age_actors, Num_multiplex, Collection

▪ Dependent attribute: oscar

▪ Constraints:

o Split 80% data for training and 20% data for testing

o Use any classification techniques, correlation and label encoding / one hot encoding techniques. Calculate the accuracy, precision, recall and F1 score for the final classification model.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Load the dataset
data = pd.read_csv("movie_classification.csv")

# Drop rows with missing values
data.dropna(inplace=True)

# Encode categorical variables using LabelEncoder
```

```
label_encoder = LabelEncoder()
data['3D_available'] = label_encoder.fit_transform(data['3D_available'])
data['Genre'] = label_encoder.fit_transform(data['Genre'])

# Split data into independent variables (X) and dependent variable (y)
X = data.drop(columns=['oscar'])
y = data['oscar']

# Split data into training and testing 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 model (e.g., Random Forest)
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Print evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
```

```
print("Recall:", recall)
print("F1 Score:", f1)
```

Task 3

Predict price of the houses in Los Angeles using the dataset “House_Price”.

▯ Independent attributes: crime_rate, resid_area, air_qual, room_num, age, airport, waterbody, rainfall, bus_ter and parks

▯ Dependent attribute: price

▯ Constraints:

o Split 80% data for training and 20% data for testing

o Use multi linear regression, correlation and label encoding / one hot encoding techniques

o Calculate the MSE for the final multi linear regression model

o Print the coefficients of slops and intercepts

```
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

# Load the dataset
data = pd.read_csv("House_Price.csv")

# Drop rows with missing values
data.dropna(inplace=True)

# Encode categorical variable (if applicable)
# For this task, 'waterbody' is a categorical variable
```

```
label_encoder = LabelEncoder()
data['waterbody'] = label_encoder.fit_transform(data['waterbody'])

# Split data into independent variables (X) and dependent variable (y)
X = data.drop(columns=['price', 'airport', 'bus_ter'])
y = data['price']

# Split data into training and testing 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 multi-linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Calculate Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)

# Print MSE
print("Mean Squared Error:", mse)

# Print coefficients of slopes and intercept
print("Coefficients of slopes:", model.coef_)
print("Intercept:", model.intercept_)
```

Task 1

Estimate the collection of movie based on the basic information using the "Movie_regression" dataset.

▯ Independent attributes: Marketing expense, Production expense, Multiplex coverage, Budget, Movie_length, Lead_ Actor_Rating, Lead_Actress_rating, Director_rating, Producer_rating, Critic_rating, Trailer_views, 3D_available, Time_taken, witter_hastags, Genre, Avg_age_actors, Num_multiplex

▯ Dependent attribute: Collection

▯ Constraints:

o Split 80% data for training and 20% data for testing

o Use multi linear regression, correlation and label encoding / one hot encoding techniques

o Calculate the MSE for the final multi linear regression model

o Print the coefficients of slops and intercepts

```
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

# Load the dataset
data = pd.read_csv("Movie_regression.csv")

# Drop rows with missing values
data.dropna(inplace=True)
```

```
# Fill missing values with the mean of each column
#data.fillna(data.mean(), inplace=True)

# Encode categorical variables (if applicable)
# For this task, 'Genre' and '3D_available' are categorical variables
label_encoder = LabelEncoder()
data['Genre'] = label_encoder.fit_transform(data['Genre'])
data['3D_available'] = label_encoder.fit_transform(data['3D_available'])

# Split data into independent variables (X) and dependent variable (y)
X = data.drop(columns=['Collection'])
y = data['Collection']

# Split data into training and testing 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 multi-linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Calculate Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)

# Print MSE
```

```
print("Mean Squared Error:", mse)


# Print coefficients of slopes and intercept
print("Coefficients of slopes:", model.coef_)
print("Intercept:", model.intercept_)
```