Machine learning

Mohammed Ashraf

Introduction

- Machine learning is a field of artificial intelligence that focuses on the development of algorithms and models that enable computers to learn from data and make predictions or decisions without being explicitly programmed. The key idea is to allow machines to automatically improve their performance on a task through experience

types of machine learning projects

Supervised Learning Regression Problem

Supervised Learning Classification Problem

Unsupervised Learning

Project(1_Regrssion)

Description

- dataset related to video game sales

Problem

- Dataset include missing values, outliers, inconsistent formatting, or the need for data preprocessing steps such as handling categorical variables or normalizing numerical features. Analyzing the data could involve exploring trends in sales over the years, identifying popular genres and we used regression analysis for predicting Global_Sales

Project(1)
Data preprocessing

Data cleaning

- By checking the missing values
- Checking the duplicated values
- Checking the outliers

Data visualization

- Finding patterns to get meaning from our data

the models we used for machine learning

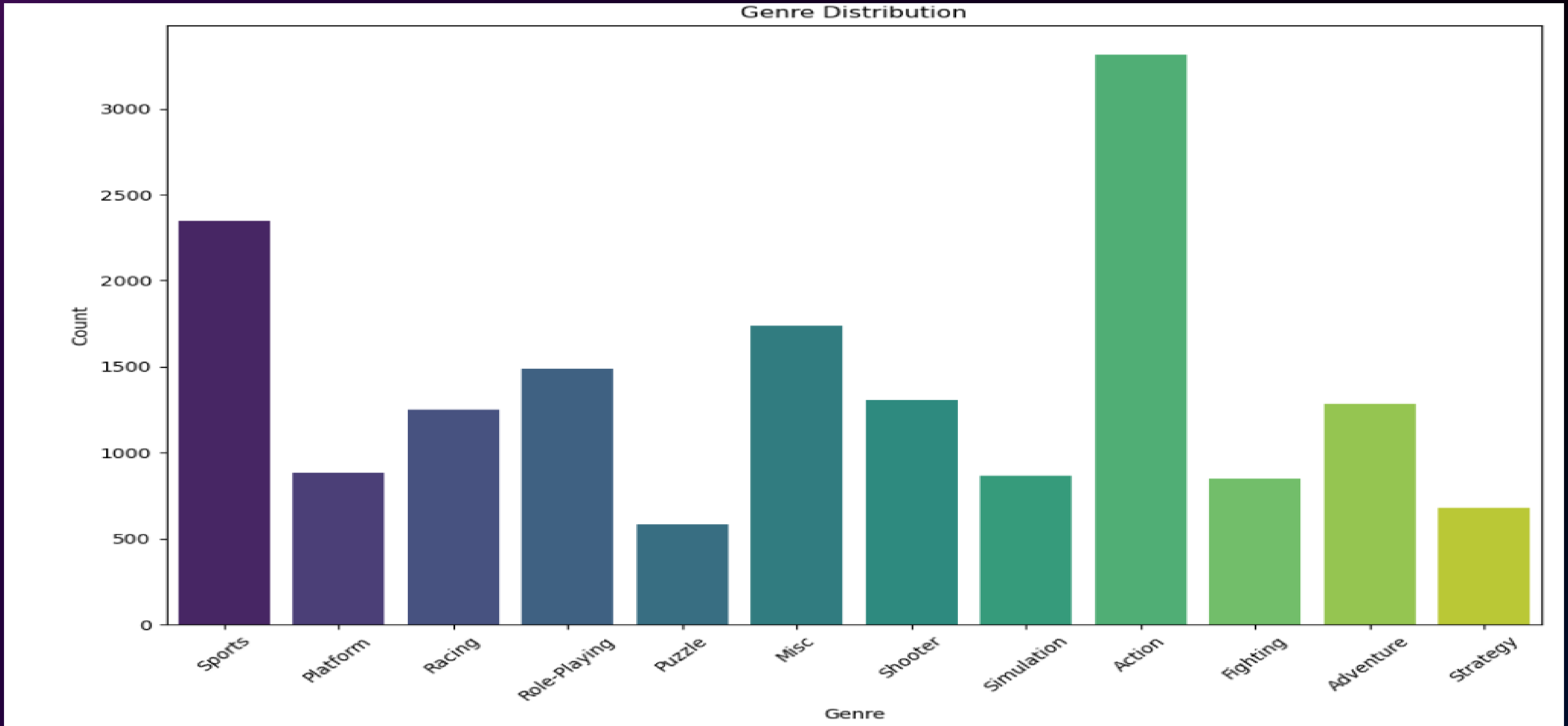
1. regression model
2. Decision Trees:
3. Random forest Regressor
4. Support Vector Regressor (SVR)
5. Gradient Boosting regressor
6. MLPRegressor

Data Before and after cleaning

```
memory usage: 1.4+ MB

In [51]: df.duplicated().sum()

Out[51]: 0

In [52]: #the total num of nulls of each column
         df.isnull().sum()

Out[52]: Rank              0
         Name              0
         Platform          0
         Year            271
         Genre             0
         Publisher        58
         NA_Sales          0
         EU_Sales          0
         JP_Sales          0
         Other_Sales       0
         Global_Sales      0
         dtype: int64
```

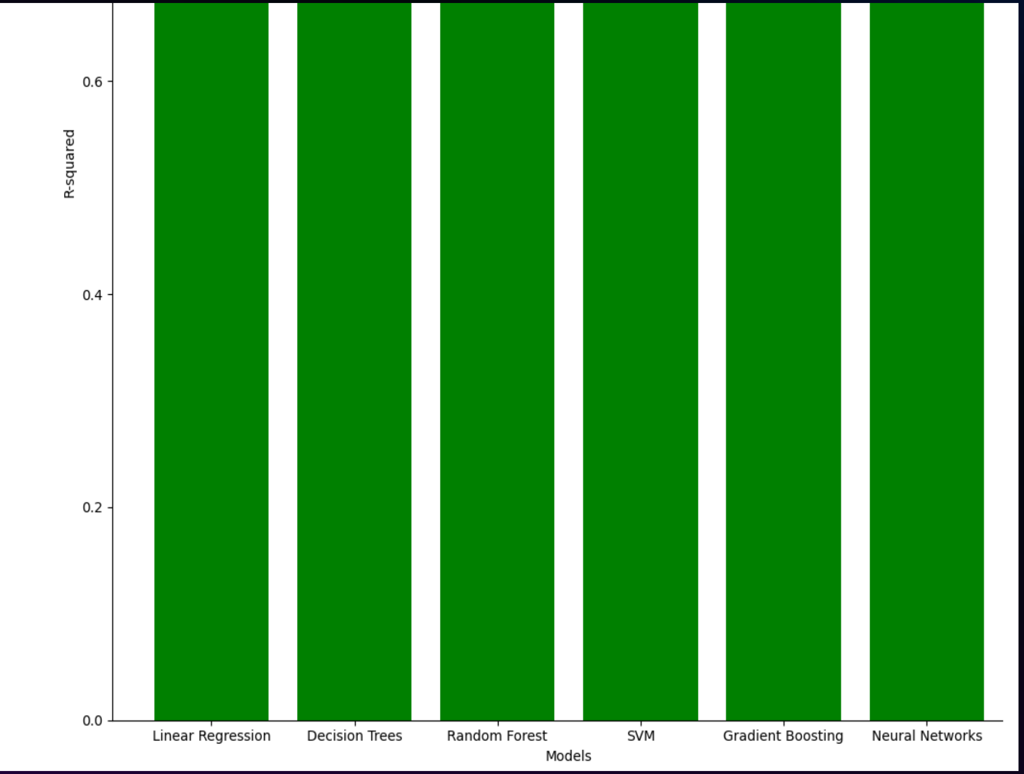
```
: df['Year'].fillna(df['Year'].mode().iloc[0], inplace=True)

: df['Publisher'].fillna(df['Publisher'].mode().iloc[0], inplace=True)

: #the total num of nulls of each column
  df.isnull().sum()

: Rank            0
  Name            0
  Platform        0
  Year            0
  Genre           0
  Publisher       0
  NA_Sales        0
  EU_Sales        0
  JP_Sales        0
  Other_Sales     0
  Global_Sales    0
  dtype: int64
```

Example of our visualization



Measuring and visualize our models

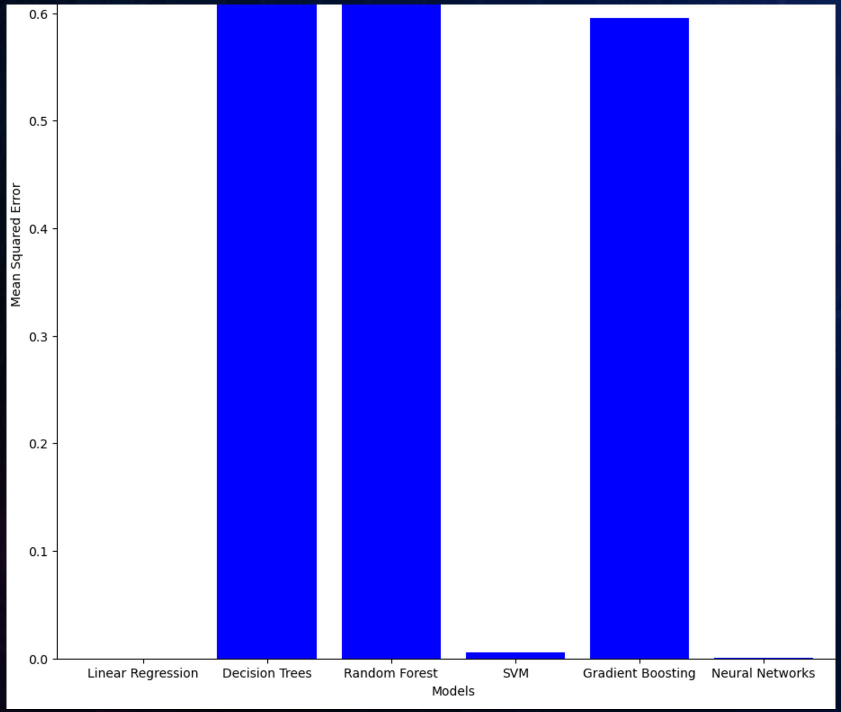
```
feature_columns = ['Year', 'NA_Sales', 'EU_Sales', 'JP_Sales', 'Other_Sales']

# Features (X)
X = df[feature_columns]

# Target variable (y)
y = df['Global_Sales']

# Split the 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 fit the linear regression model

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

LinearRegression
LinearRegression()

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

# Evaluate the model
mse_linear_reg = mean_squared_error(y_test, y_pred)
r2_linear_reg = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse_linear_reg}')
print(f'R-squared: {r2_linear_reg}')

Mean Squared Error: 2.7401694787214883e-05
R-squared: 0.9999934779050468
```




```
#Gradient Boosting regressor:
from sklearn.ensemble import GradientBoostingRegressor

# Create a gradient boosting model
gradient_boosting = GradientBoostingRegressor(n_estimators=100, random_state=42)

# Train the model
gradient_boosting.fit(X_train, y_train)

GradientBoostingRegressor
GradientBoostingRegressor(random_state=42)

# Make predictions
predictions = gradient_boosting.predict(X_test)

# Evaluate the model
mse_gradient_boosting = mean_squared_error(y_test, predictions)
r2_gradient_boosting = r2_score(y_test, predictions)

print(f'Mean Squared Error: {mse_gradient_boosting}')
print(f'R-squared: {r2_gradient_boosting}')

Mean Squared Error: 0.595694701016712
R-squared: 0.8582139742340673
```

Sample of
Comparison between models

model	mse	Rmse
Linear Regression	2.740169478721488	0.9999934779050468
DecisionTreeRegressorr	0.7353510209920812	0.8249732646077993
SVR	0.00545995208480858	0.9987004334507993
GradientBoostingRegressor	0.595694701016712	0.8582139742340673



Project(2_classification)

Description

- data set related to employees.

Problem

- Dataset include missing values, outliers, inconsistent formatting, or the need for data preprocessing steps such as handling categorical variables or normalizing numerical features. Analyzing the data could involve exploring trends so we used Classification models for predicting LeaveOrNot

Project(2)
Data preprocessing

Data cleaning

- By checking the missing values
- Checking the duplicated values
- Checking the outliers

Data visualization

- Finding patterns to get meaning from our data

the models we used for machine learning

1. LogisticRegression
2. DecisionTreeClassifier
3. RandomForestClassifier
4. SVC
5. KNeighborsClassifier
6. MLPClassifier

Data Before and after cleaning

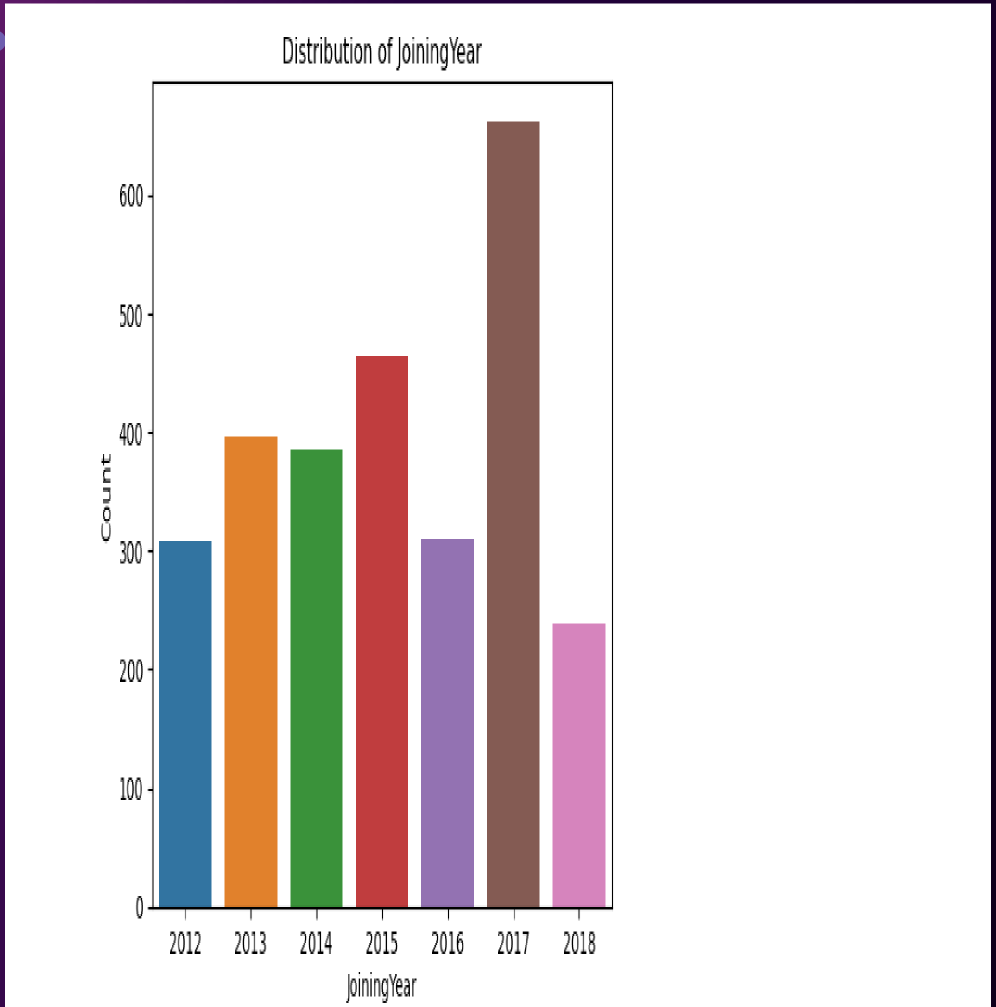
```
In [32]: df.duplicated().sum()
Out[32]: 1889

In [33]: df = df.drop_duplicates()

In [34]: df.duplicated().sum()
Out[34]: 0

In [35]: #the total num of nulls of each column
         df.isnull().sum()
Out[35]: Education                    0
         JoiningYear                  0
         City                         0
         PaymentTier                  0
         Age                          0
         Gender                       0
         EverBenched                  0
         ExperienceInCurrentDomain    0
         LeaveOrNot                   0
         dtype: int64
```

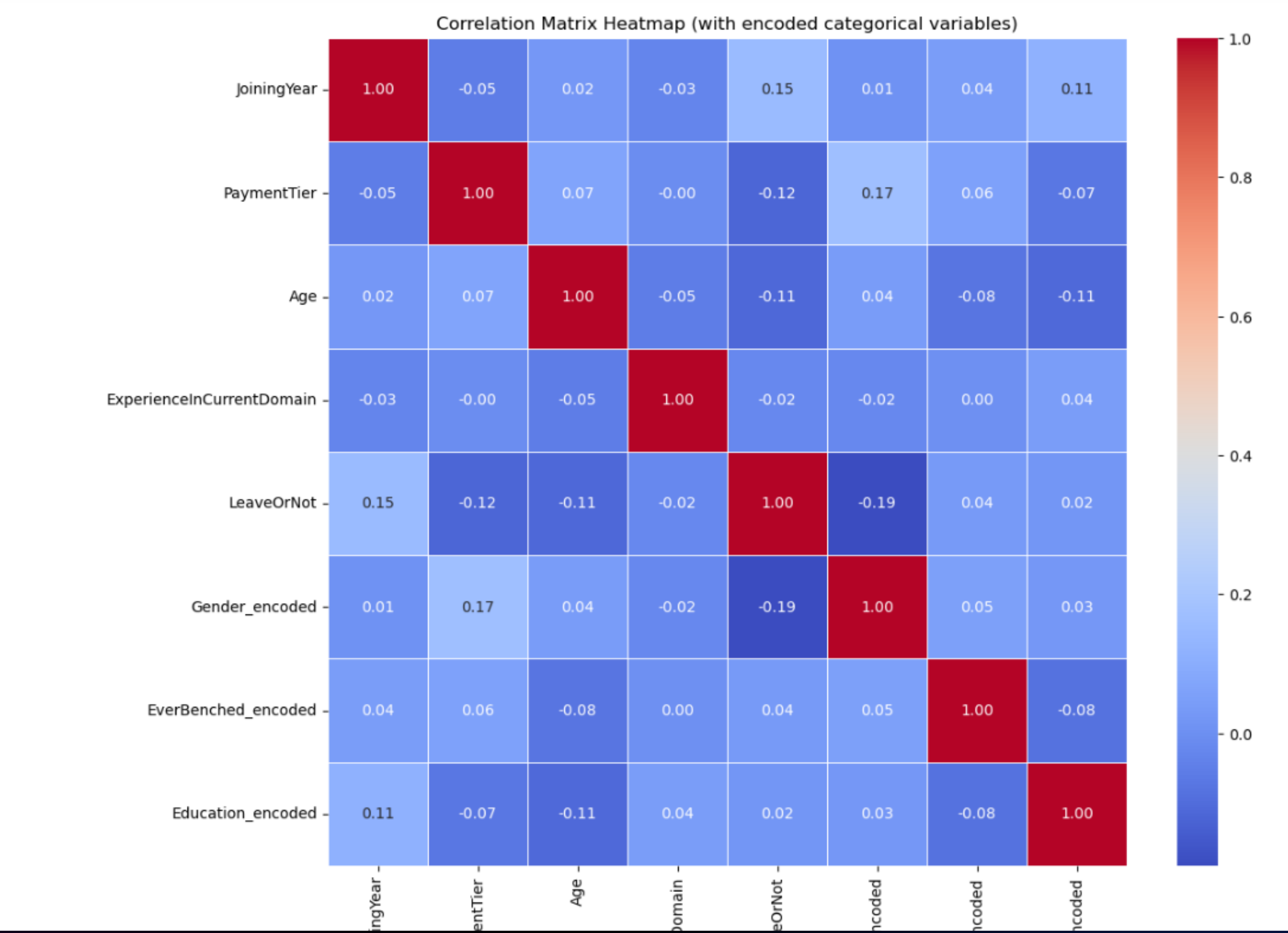
```
#getting the columns name
df.columns

Index(['Education', 'JoiningYear', 'City', 'PaymentTier', 'Age', 'Gender',
       'EverBenched', 'ExperienceInCurrentDomain', 'LeaveOrNot'],
      dtype='object')

df
```

	Education	JoiningYear	City	PaymentTier	Age	Gender	EverBenched	ExperienceInCurrentDomain	LeaveOrNot
0	Bachelors	2017	Bangalore	3	34	Male	No	0	0
1	Bachelors	2013	Pune	1	28	Female	No	3	1
2	Bachelors	2014	New Delhi	3	38	Female	No	2	0
3	Masters	2016	Bangalore	3	27	Male	No	5	1
4	Masters	2017	Pune	3	24	Male	Yes	2	1
...	...	...	...	...	...	...	...	...	...
4645	Masters	2017	Pune	2	31	Female	No	2	0
4647	Bachelors	2016	Pune	3	30	Male	No	2	0
4649	Masters	2013	Pune	2	37	Male	No	2	1
4650	Masters	2018	New Delhi	3	27	Male	No	5	1
4651	Bachelors	2012	Bangalore	3	30	Male	Yes	2	0

2764 rows × 9 columns

Example of our visualization





Sample of our models

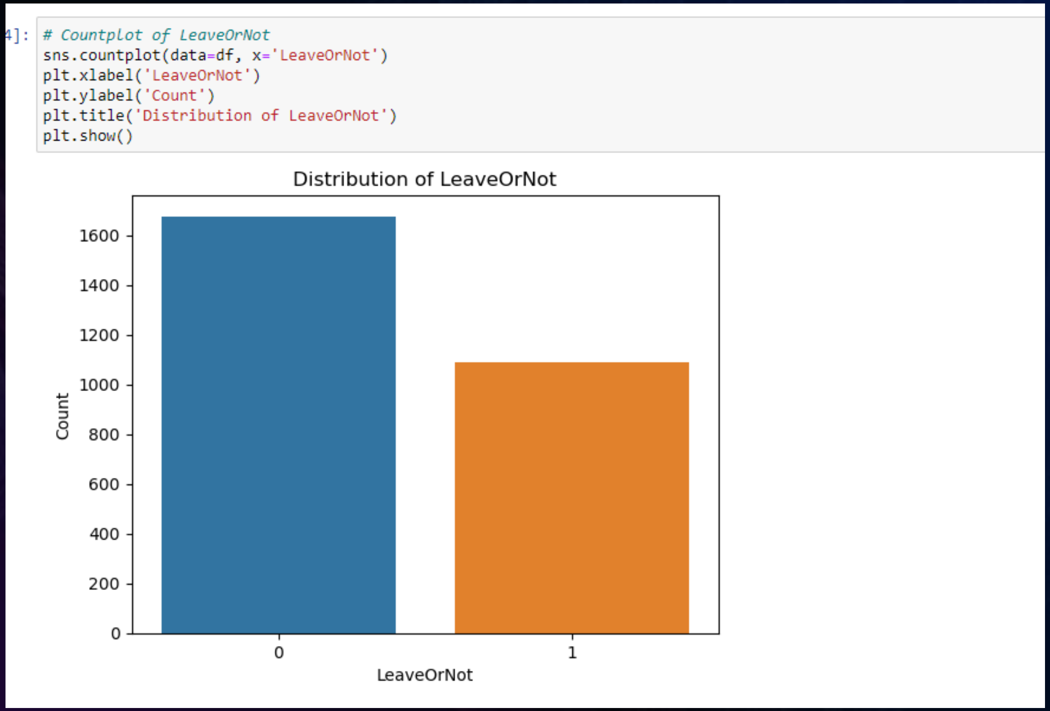
```
# Assuming 'Education', 'JoiningYear', 'City', 'PaymentTier', 'Age', 'Gender', 'EverBenched', and 'ExperienceInCurrentDomain' are
X = df[['Education_encoded', 'JoiningYear', 'PaymentTier', 'Age', 'Gender_encoded', 'EverBenched_encoded', 'ExperienceInCurrentDo
y = df['LeaveOrNot']  # Replace 'Target' with the actual name of your target column

[58]: # Split the 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)

[59]: # Standardize features by removing the mean and scaling to unit variance
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

[60]: # Create a logistic regression model
model = LogisticRegression(random_state=42)

[61]: # Train the model
model.fit(X_train_scaled, y_train)

[61]: LogisticRegression
LogisticRegression(random_state=42)

[62]: # Make predictions on the test set
y_pred = model.predict(X_test_scaled)

[63]: # Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

[64]: print(f'Accuracy: {accuracy}')

Accuracy: 0.6582278481012658
```

```
In [65]: print(f'Confusion Matrix:\n{conf_matrix}')

Confusion Matrix:
[[287  46]
 [143  77]]

In [66]: print(f'Classification Report:\n{classification_rep}')

Classification Report:
              precision    recall  f1-score   support

           0       0.67      0.86      0.75       333
           1       0.63      0.35      0.45       220

    accuracy                           0.66       553
   macro avg       0.65      0.61      0.60       553
weighted avg       0.65      0.66      0.63       553

In [67]: print("Training set class distribution:")
print(y_train.value_counts())

print("\nTest set class distribution:")
print(y_test.value_counts())

Training set class distribution:
LeaveOrNot
0    1343
1     868
Name: count, dtype: int64

Test set class distribution:
LeaveOrNot
0    333
1    220
Name: count, dtype: int64
```

```
#DecisionTreeClassifier

# Assuming 'Education', 'JoiningYear', 'City', 'PaymentTier', 'Age', 'Gender', 'EverBenched', and 'ExperienceInCurrentDomain' are
X = df[['Education_encoded', 'JoiningYear', 'PaymentTier', 'Age', 'Gender_encoded', 'EverBenched_encoded', 'ExperienceInCurrentDo
y = df['LeaveOrNot']  # Replace 'Target' with the actual name of your target column

# Split the data into training and testing sets
X_train, X_test, y_train_d, y_test_d = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a decision tree model
model = DecisionTreeClassifier(random_state=42)

# Train the model
model.fit(X_train, y_train_d)

DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)

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

# Evaluate the model
accuracy = accuracy_score(y_test_d, y_pred)
conf_matrix = confusion_matrix(y_test_d, y_pred)
classification_rep = classification_report(y_test_d, y_pred)

print(f'Accuracy: {accuracy}')

Accuracy: 0.6853526220614828

print(f'Confusion Matrix:\n{conf_matrix}')

Confusion Matrix:
[[256  77]
 [ 97 123]]
```

Sample of Comparison between models

model	Accuracy
Logistic Regression	0.6582278481012658
DecisionTreeClassifier	0.6853526220614828
RandomForestClassifier	0.6980108499095841
MLPClassifier	0.7775768535262206



Machine learning

Project(3_new techniques to classify)

Description

- **dataset related to diabetes patients**

Problem

- **Dataset include missing values, outliers, inconsistent formatting, or the need for data preprocessing steps such as handling categorical variables or normalizing numerical features. Analyzing the data could involve exploring trends so we used** unsupervised learning techniques on it like Principal Component Analysis and Independent Component Analysis,...etc.

Project(3)
Data preprocessing

Data cleaning

- By checking the missing values
- Checking the duplicated values
- Checking the outliers

Data visualization

- Finding patterns to get meaning from our data

the models we used for machine learning

1. **Principal Component Analysis (PCA)**
2. **Independent Component Analysis (ICA)**
3. **DBSCAN**
4. **t-Distributed Stochastic Neighbor Embedding (t- SNE)**

Principal Component Analysis (PCA):

Benefit: Dimensionality Reduction
PCA helps in reducing the dimensionality of a dataset by transforming it into a new coordinate system where the data's variability is maximized along the principal components. This simplifies the analysis and visualization of high-dimensional data, retaining the most important information.

Independent Component Analysis (ICA):
Benefit: Source Separation
ICA is particularly useful when dealing with mixed signals or data sources. It helps separate the original, independent sources from their linear mixtures. This is valuable in fields like signal processing and neuroscience, where distinguishing between contributing factors is crucial.

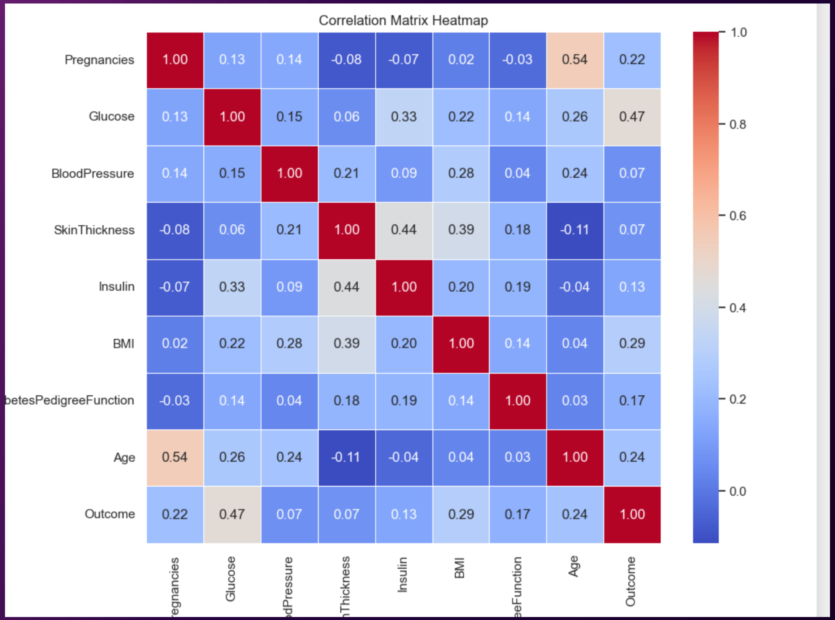
DBSCAN (Density-Based Spatial Clustering of Applications with Noise):
Benefit: Robust Clustering with Noise Handling
DBSCAN is effective in identifying clusters in data based on density, making it robust to outliers and noise. It can discover clusters of arbitrary shapes and is less sensitive to the initial choice of parameters. This makes it suitable for datasets with irregular cluster shapes and varying densities.

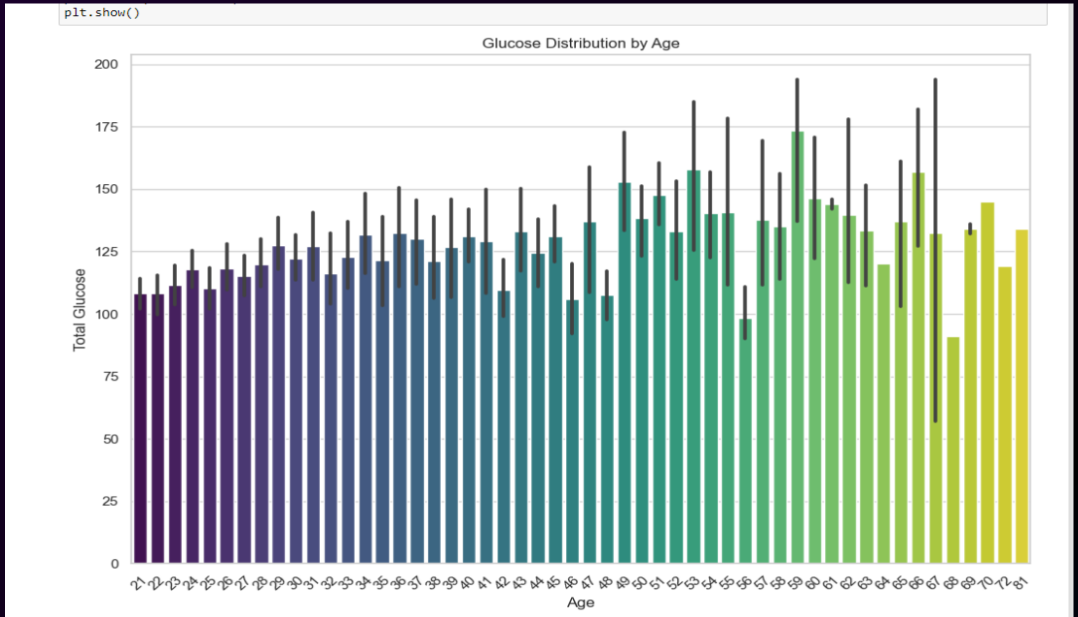
t-Distributed Stochastic Neighbor Embedding (t-SNE):
Benefit: Non-linear Dimensionality Reduction
t-SNE is particularly adept at capturing non-linear relationships in high-dimensional data, making it useful for visualizing complex structures in lower-dimensional space. It excels at preserving local relationships between data points, making it a powerful tool for exploring and interpreting intricate patterns in the data.

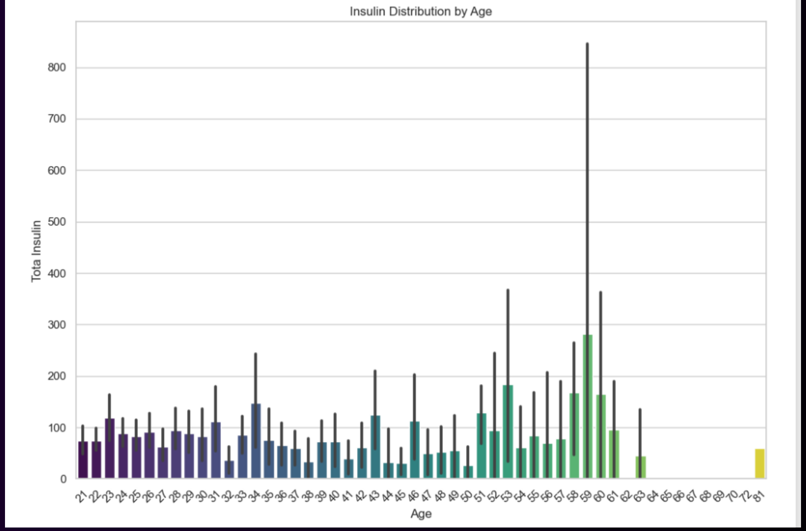
Check our data is cleaned

```
In [37]: df.duplicated().sum()

Out[37]: 0

In [38]: #the total num of nulls of each column
         df.isnull().sum()

Out[38]: Pregnancies                 0
         Glucose                     0
         BloodPressure               0
         SkinThickness               0
         Insulin                     0
         BMI                         0
         DiabetesPedigreeFunction    0
         Age                         0
         Outcome                     0
         dtype: int64
```

Example of our visualization



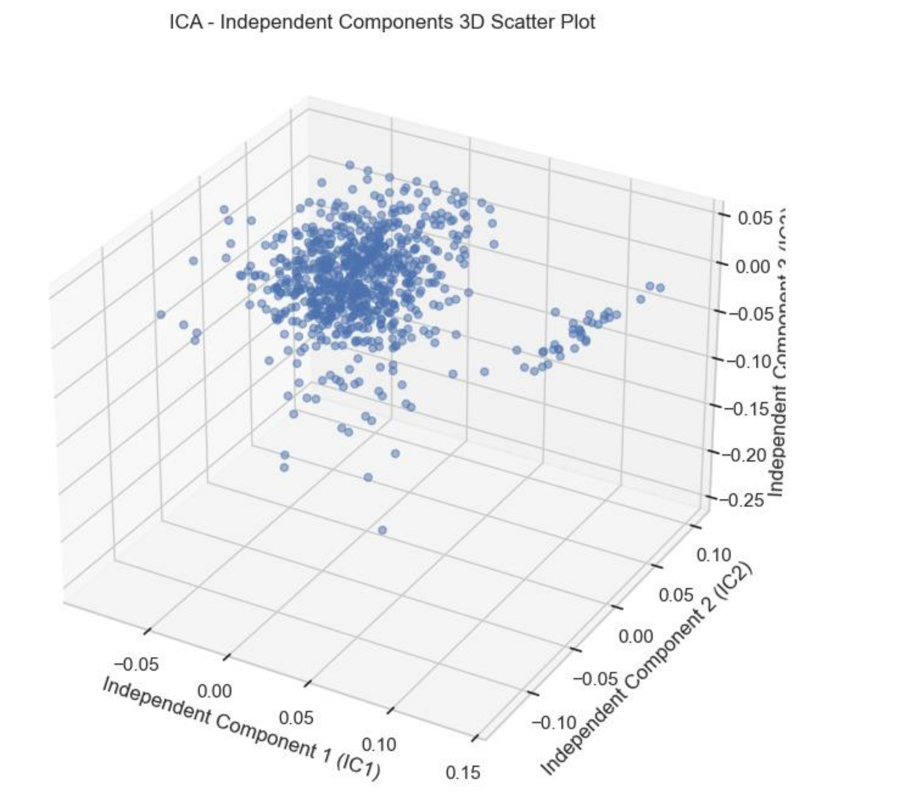


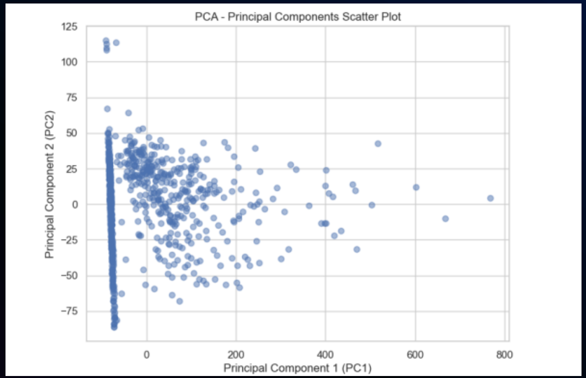


Sample of our models

```
In [88]: #DBSCAN:

#library for it
from sklearn.cluster import DBSCAN

features = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']
X = df[features]

# Apply DBSCAN
dbscan = DBSCAN(eps=0.5, min_samples=5)
df['cluster'] = dbscan.fit_predict(X)
print(df['cluster'])

0     -1
1     -1
2     -1
3     -1
4     -1
      ..
763   -1
764   -1
765   -1
766   -1
767   -1
Name: cluster, Length: 768, dtype: int64
```

```
#Principal Component Analysis (PCA):

# Select features
features = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']
X = df[features]

# Apply PCA for dimensionality reduction
pca = PCA(n_components=2)
components = pca.fit_transform(X)

# Create a new DataFrame with the principal components
df_pca = pd.DataFrame(data=components, columns=['PC1', 'PC2'])
print(df_pca)

           PC1        PC2
0   -75.717197 -35.940785
1   -82.360030  28.913854
2   -74.628227 -67.925129
3    11.082730  34.880106
4    89.747409  -2.756199
..         ...        ...
763  99.231740  25.104141
764 -78.646635  -7.665117
765  32.117209   3.359444
766 -80.209607 -14.204597
767 -81.307419  21.621617

[768 rows x 2 columns]
```





Project(4_Clothes classification)

Description

- dataset is related to values of pixels of images

Problem

- We need to turn our model from 1D to 2D to collect pixels so we will train our data on CNN model to classify the types of the clothes based on the nine classes to classify the type of the clothes

Project(4)
Data preprocessing

Data Cleaning

- By checking the missing values
- Checking the duplicated values
- Checking the outliers

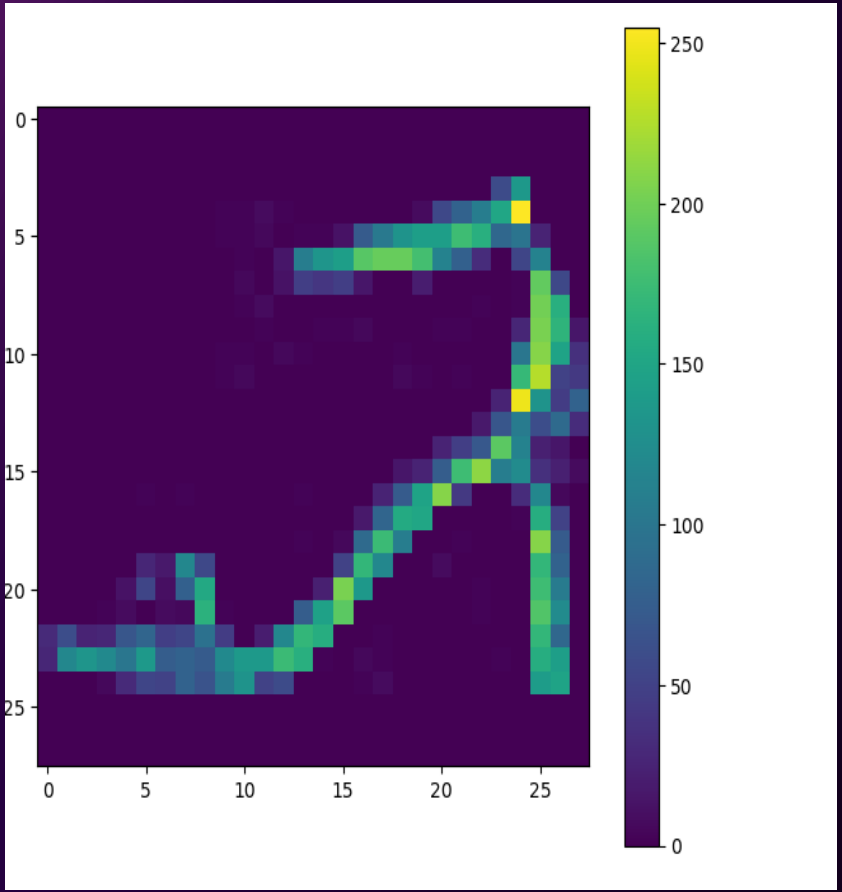
Data visualization

- Visualization of the distribution of classes (e.g., types of clothing such as shirts, pants, shoes) helps to understand the balance or imbalance in the dataset. Bar charts or pie charts can be useful for this.

the models we used for machine learning

Building model CNN

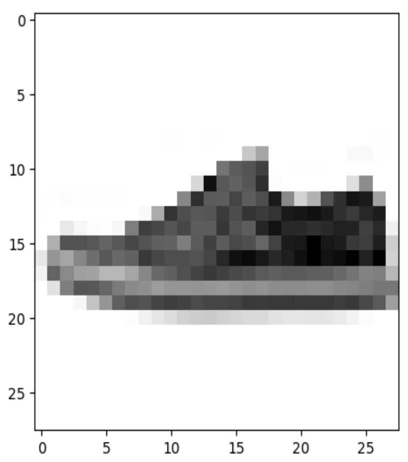
Example of our visualization



Example of our visualization

```
plt.imshow(X_train.iloc[1].values.reshape(28, 28),cmap = 'Greys')
```

<matplotlib.image.AxesImage at 0x274f5aa8a90>




```
import numpy as np
# Initialize an empty list to store reshaped images
x_train = []
# Iterate through each row in X_train and reshape
for i in range(len(X_train)):
    x_train.append(X_train.iloc[i].values.reshape(28, 28))

# Convert the list to a NumPy array
reshaped_images_train = np.array(x_train)

# The resulting array reshaped_images_train
print(reshaped_images_train.shape)
```

(48000, 28, 28)

```
import numpy as np
# Initialize an empty list to store reshaped images
x_test = []
# Iterate through each row in X_train and reshape
for i in range(len(X_test)):
    x_test.append(X_test.iloc[i].values.reshape(28, 28))

# Convert the list to a NumPy array
reshaped_images_test = np.array(x_test)

# The resulting array reshaped_images_train
print(reshaped_images_test.shape)
```

(12000, 28, 28)

```
model = keras.models.Sequential([
    keras.layers.Conv2D(filters=64, kernel_size=3, strides=(1, 1), padding='vali
    keras.layers.MaxPooling2D(pool_size=(2, 2)),

    keras.layers.Conv2D(filters=128, kernel_size=3, strides=(2, 2), padding='sam
    keras.layers.MaxPooling2D(pool_size=(2, 2)),

    keras.layers.Conv2D(filters=64, kernel_size=3, strides=(2, 2), padding='same
    keras.layers.MaxPooling2D(pool_size=(2, 2)),

    keras.layers.Flatten(),
    keras.layers.Dense(units=128, activation='relu'),
    keras.layers.Dropout(0.25),
    keras.layers.Dense(units=256, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(units=256, activation='relu'),
    keras.layers.Dropout(0.25),
    keras.layers.Dense(units=128, activation='relu'),
    keras.layers.Dropout(0.10),
    keras.layers.Dense(units=10, activation='softmax')  # Output layer
])
```

```
keras.utils.plot_model(model, to_file='model_plot.png', show_shapes=True, show_l
```

```
plt.figure(figsize=(16, 16))
j = 1
for i in np.random.randint(0, len(y_train), 25):
    plt.subplot(5, 5, j)
    j += 1
    if i < len(reshaped_images_train):  # Check if the index is within the valid range
        plt.imshow(reshaped_images_train[i], cmap='Greys')
        plt.axis('off')
        plt.title('{} / {}'.format(class_names[y_train.iloc[i]], y_train.iloc[i]))

plt.show()
```

The result and the accuracy


```
]: cr = classification_report (y_test,y_pred_labels,target_names = class_labels)
print(cr)
```

```
              precision    recall  f1-score   support

 T-shirt/top       0.84      0.85      0.85      1232
     Trouser       0.97      0.98      0.98      1174
    Pullover       0.84      0.88      0.86      1200
       Dress       0.92      0.89      0.90      1242
        Coat       0.83      0.85      0.84      1185
      Sandal       0.99      0.95      0.97      1141
       Shirt       0.75      0.71      0.73      1243
    Sneakers       0.94      0.98      0.96      1224
         Bag       0.98      0.98      0.98      1149
  Ankle boot       0.97      0.97      0.97      1210

    accuracy                           0.90     12000
   macro avg       0.90      0.90      0.90     12000
weighted avg       0.90      0.90      0.90     12000
```

Thank you

Mohammed Ashraf

4210395