elevvo-ml-intern

August 14, 2025

https://colab.research.google.com/drive/1WgFGPhsLwmuWpaHR-Pj3X_QbWERf1Nfl?usp=sharing

Github Repository

[6]:
```
%cd https://github.com/zoya4477/ML-intern.git
!git clone
!git config --global user.email "zoyahafeez785@gmail.com"
!git config --global user.name "zoya4477"
```

```
[Errno 2] No such file or directory: 'https://github.com/zoya4477/ML-intern.git'
/content
fatal: You must specify a repository to clone.

usage: git clone [<options>] [--] <repo> [<dir>]

    -v, --verbose         be more verbose
    -q, --quiet           be more quiet
    --progress            force progress reporting
    --reject-shallow      don't clone shallow repository
    -n, --no-checkout     don't create a checkout
    --bare                create a bare repository
    --mirror              create a mirror repository (implies bare)
    -l, --local           to clone from a local repository
    --no-hardlinks        don't use local hardlinks, always copy
    -s, --shared          setup as shared repository
    --recurse-submodules[=<pathspec>]
                          initialize submodules in the clone
    --recursive …       alias of --recurse-submodules
    -j, --jobs <n>        number of submodules cloned in parallel
    --template <template-directory>
                          directory from which templates will be used
    --reference <repo>    reference repository
    --reference-if-able <repo>
                          reference repository
    --dissociate          use --reference only while cloning
    -o, --origin <name>   use <name> instead of 'origin' to track upstream
    -b, --branch <branch>
                          checkout <branch> instead of the remote's HEAD
    -u, --upload-pack <path>
```

```
                          path to git-upload-pack on the remote
    --depth <depth>       create a shallow clone of that depth
    --shallow-since <time>
                          create a shallow clone since a specific time
    --shallow-exclude <revision>
                          deepen history of shallow clone, excluding rev
    --single-branch       clone only one branch, HEAD or --branch
    --no-tags             don't clone any tags, and make later fetches not to
follow them
    --shallow-submodules  any cloned submodules will be shallow
    --separate-git-dir <gitdir>
                          separate git dir from working tree
    -c, --config <key=value>
                          set config inside the new repository
    --server-option <server-specific>
                          option to transmit
    -4, --ipv4            use IPv4 addresses only
    -6, --ipv6            use IPv6 addresses only
    --filter <args>       object filtering
    --remote-submodules   any cloned submodules will use their remote-tracking
branch
    --sparse              initialize sparse-checkout file to include only files
at root
```

1 Task 1: Student Score Prediction

[ ]:
```
#Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import seaborn as sns
```

[ ]:
```
#Load Dataset
df = pd.read_csv("/content/archive.zip")
df.head()
```

[ ]:
```
   Hours_Studied  Attendance Parental_Involvement Access_to_Resources  \
0             23          84                  Low                High
1             19          64                  Low              Medium
2             24          98               Medium              Medium
3             29          89                  Low              Medium
4             19          92               Medium              Medium
```

```
  Extracurricular_Activities  Sleep_Hours  Previous_Scores Motivation_Level  \
0                         No            7               73              Low
1                         No            8               59              Low
2                        Yes            7               91           Medium
3                        Yes            8               98           Medium
4                        Yes            6               65           Medium

  Internet_Access  Tutoring_Sessions Family_Income Teacher_Quality  \
0             Yes                  0           Low          Medium
1             Yes                  2        Medium          Medium
2             Yes                  2        Medium          Medium
3             Yes                  1        Medium          Medium
4             Yes                  3        Medium            High

  School_Type Peer_Influence  Physical_Activity Learning_Disabilities  \
0      Public       Positive                  3                    No
1      Public       Negative                  4                    No
2      Public        Neutral                  4                    No
3      Public       Negative                  4                    No
4      Public        Neutral                  4                    No

  Parental_Education_Level Distance_from_Home  Gender  Exam_Score
0              High School               Near    Male          67
1                  College           Moderate  Female          61
2             Postgraduate               Near    Male          74
3              High School           Moderate    Male          71
4                  College               Near  Female          70
```

Data Cleaning

[ ]:
```python
# Check for missing values
print(df.isnull().sum())

# Drop rows with nulls or fill them
df = df.dropna()  # or df.fillna(method='ffill', inplace=True)

# View data types and check unique values
df.info()
```

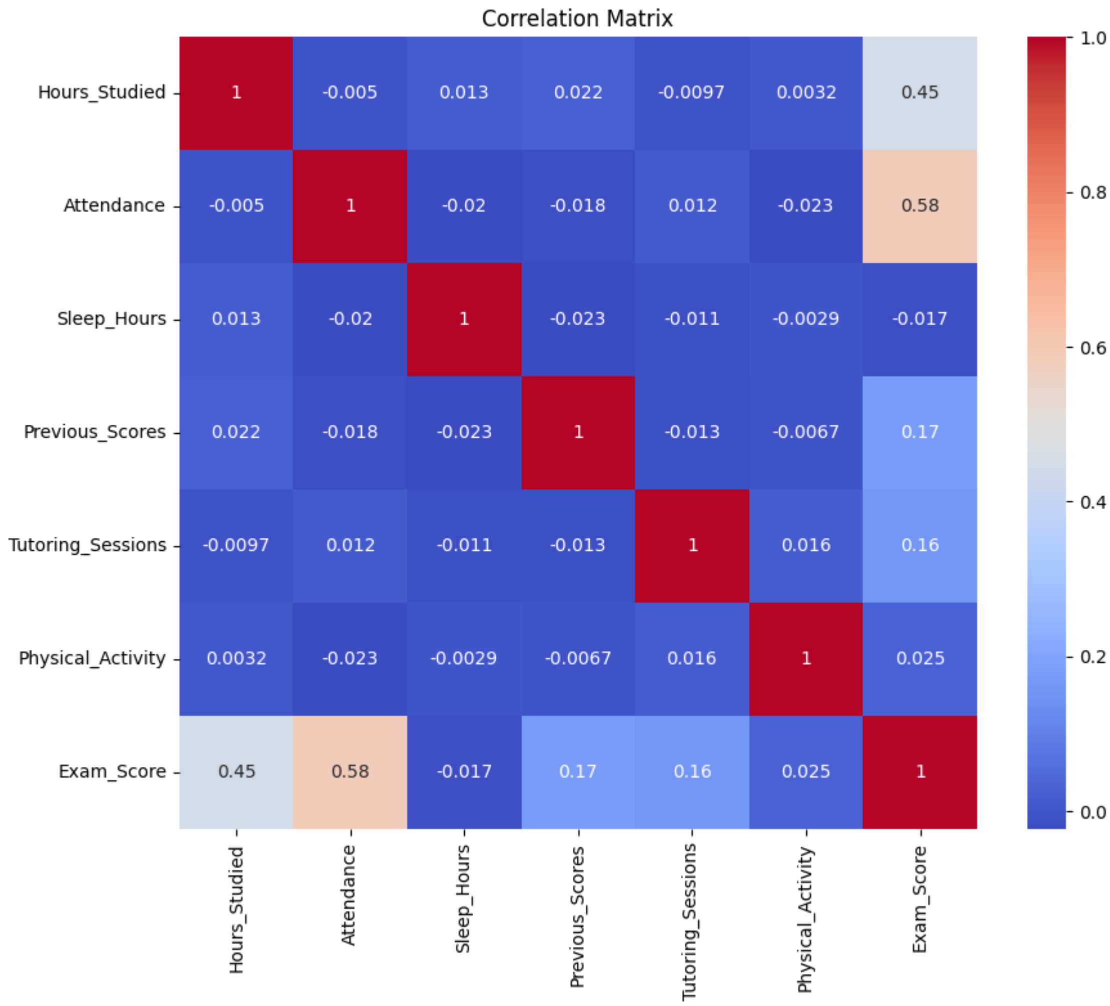
```
Hours_Studied                 0
Attendance                    0
Parental_Involvement          0
Access_to_Resources           0
Extracurricular_Activities    0
Sleep_Hours                   0
Previous_Scores               0
Motivation_Level              0
```

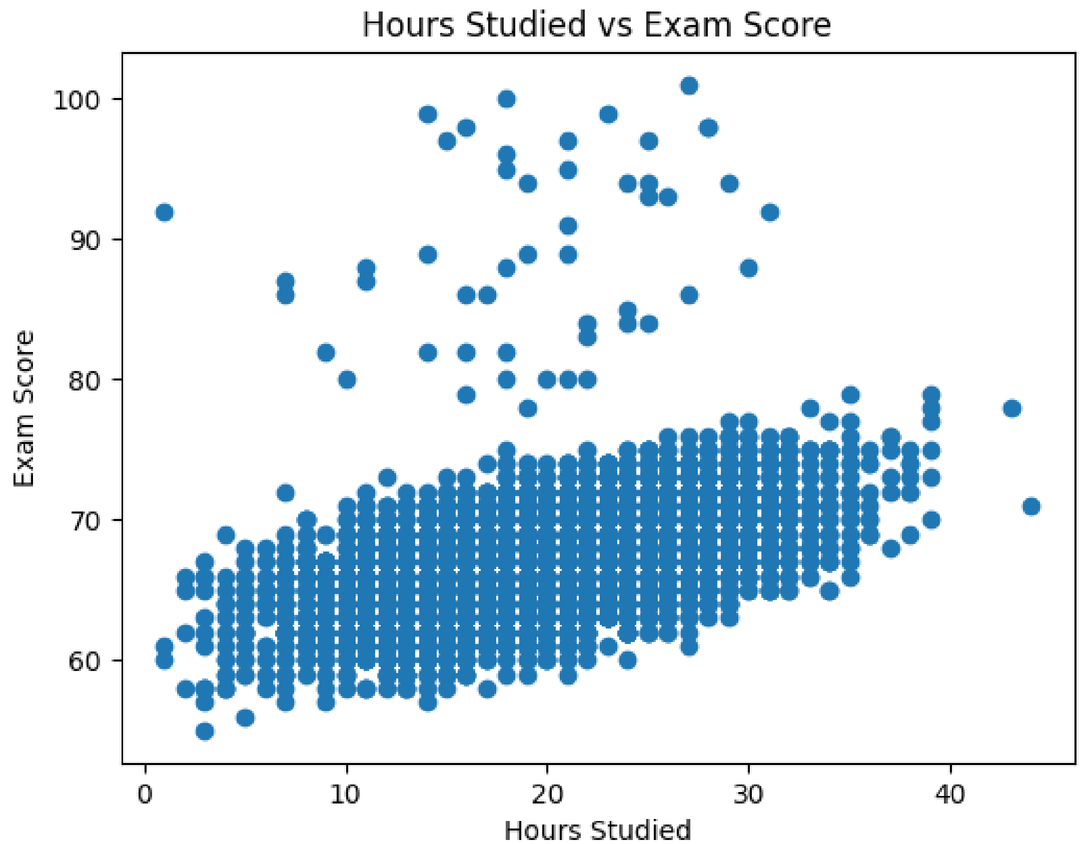
```
Internet_Access               0
Tutoring_Sessions             0
Family_Income                 0
Teacher_Quality              78
School_Type                   0
Peer_Influence                0
Physical_Activity             0
Learning_Disabilities         0
Parental_Education_Level     90
Distance_from_Home           67
Gender                        0
Exam_Score                    0
dtype: int64
<class 'pandas.core.frame.DataFrame'>
Index: 6378 entries, 0 to 6606
Data columns (total 20 columns):
 #   Column                      Non-Null Count  Dtype 
---  ------                      --------------  ----- 
 0   Hours_Studied               6378 non-null   int64 
 1   Attendance                  6378 non-null   int64 
 2   Parental_Involvement        6378 non-null   object
 3   Access_to_Resources         6378 non-null   object
 4   Extracurricular_Activities  6378 non-null   object
 5   Sleep_Hours                 6378 non-null   int64 
 6   Previous_Scores             6378 non-null   int64 
 7   Motivation_Level            6378 non-null   object
 8   Internet_Access             6378 non-null   object
 9   Tutoring_Sessions           6378 non-null   int64 
 10  Family_Income               6378 non-null   object
 11  Teacher_Quality             6378 non-null   object
 12  School_Type                 6378 non-null   object
 13  Peer_Influence              6378 non-null   object
 14  Physical_Activity           6378 non-null   int64 
 15  Learning_Disabilities       6378 non-null   object
 16  Parental_Education_Level    6378 non-null   object
 17  Distance_from_Home          6378 non-null   object
 18  Gender                      6378 non-null   object
 19  Exam_Score                  6378 non-null   int64 
dtypes: int64(7), object(13)
memory usage: 1.0+ MB
```

Basic Data Visualization

[ ]:
```
# Correlation matrix
plt.figure(figsize=(10, 8))
# Select only numerical columns for correlation calculation
numerical_df = df.select_dtypes(include=[np.number])
sns.heatmap(numerical_df.corr(), annot=True, cmap='coolwarm')
```

```
plt.title('Correlation Matrix')
plt.show()

# Scatter plot: study time vs final grade
plt.scatter(df['Hours_Studied'], df['Exam_Score'])
plt.xlabel('Hours Studied')
plt.ylabel('Exam Score')
plt.title('Hours Studied vs Exam Score')
plt.show()
```





Feature Selection

```
[ ]: # Let's use Hours_Studied to predict Exam_Score
X = df[['Hours_Studied']]  # Features
y = df['Exam_Score']             # Target variable
```

Train Test Split

```
[ ]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
  ↪random_state=42)
```

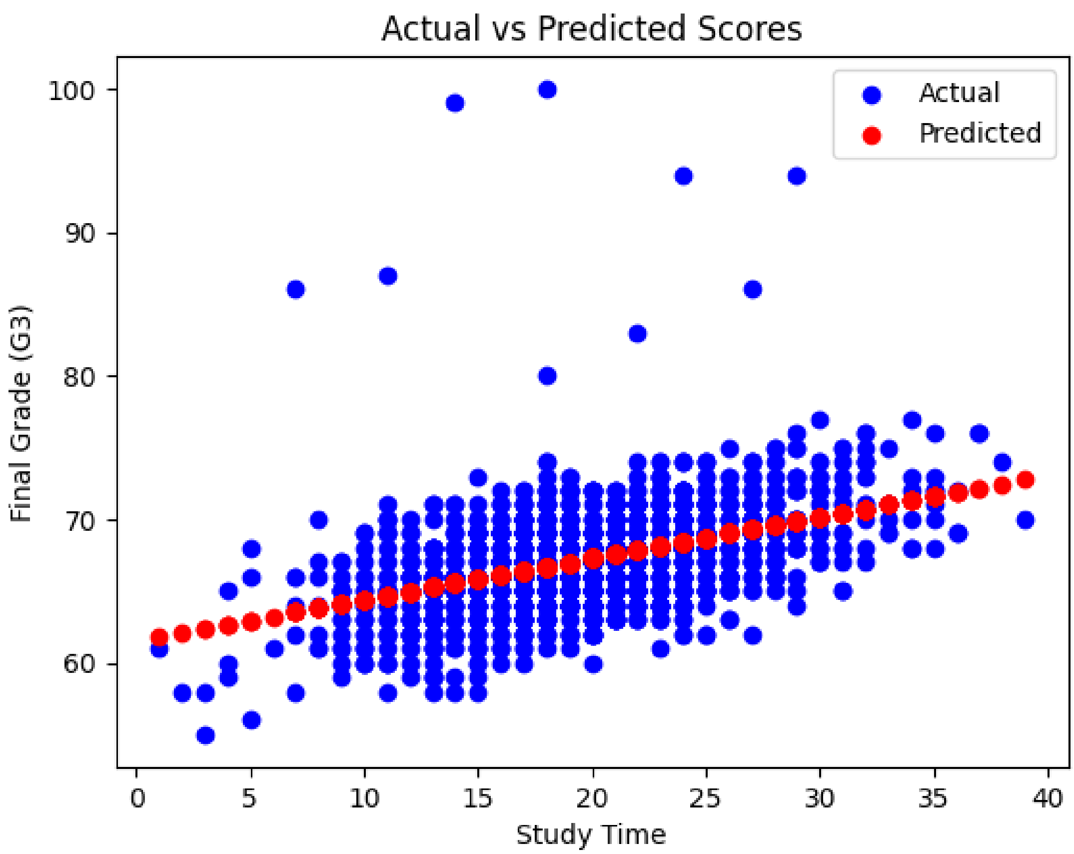
Train Linear Regression Model

```
[ ]: lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
y_pred = lr_model.predict(X_test)
```

Visualize Prediction

```
[ ]: plt.scatter(X_test, y_test, color='blue', label='Actual')
plt.scatter(X_test, y_pred, color='red', label='Predicted')
```

```
plt.xlabel('Study Time')
plt.ylabel('Final Grade (G3)')
plt.legend()
plt.title('Actual vs Predicted Scores')
plt.show()
```



Model Evaluation

[ ]:
```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse:.2f}")
print(f"R-squared: {r2:.2f}")
```

```
Mean Squared Error: 12.35
R-squared: 0.21
```

Polynomial Regression

[ ]:
```
poly = PolynomialFeatures(degree=3)
X_poly = poly.fit_transform(X)
```

```
X_train_p, X_test_p, y_train_p, y_test_p = train_test_split(X_poly, y,␣
↪test_size=0.2, random_state=42)

poly_model = LinearRegression()
poly_model.fit(X_train_p, y_train_p)
y_pred_poly = poly_model.predict(X_test_p)

# Evaluation
mse_poly = mean_squared_error(y_test_p, y_pred_poly)
r2_poly = r2_score(y_test_p, y_pred_poly)

print(f"Polynomial MSE: {mse_poly:.2f}")
print(f"Polynomial R²: {r2_poly:.2f}")
```

```
Polynomial MSE: 12.37
Polynomial R²: 0.20
```

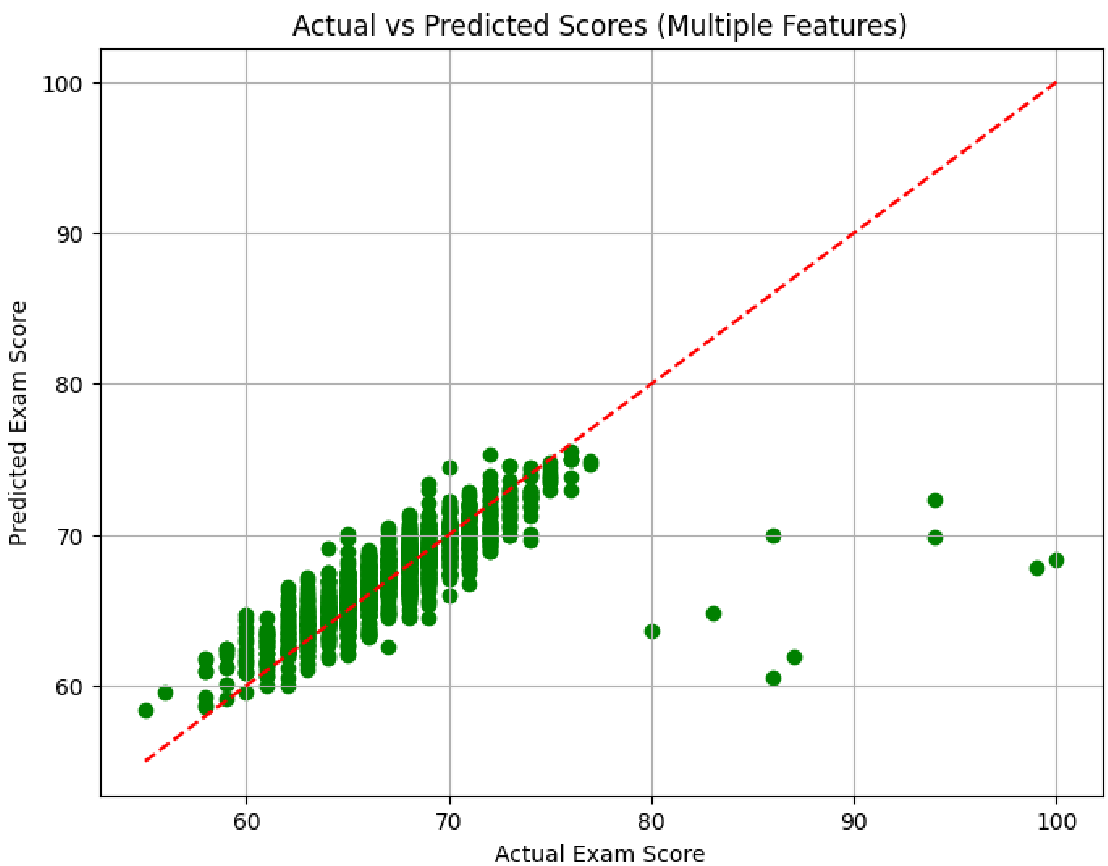
Feature Engineering (Multiple Regression)

Try experimenting with different feature combinations (e.g., removing or adding features like sleep, participation, etc.)

[ ]:
```
# Experimenting with different feature combinations
features_experiment = ['Hours_Studied', 'Attendance', 'Previous_Scores',␣
↪'Sleep_Hours', 'Physical_Activity']
X_experiment = df[features_experiment]
y_experiment = df['Exam_Score']

# Split the data with the new features
X_train_exp, X_test_exp, y_train_exp, y_test_exp =␣
↪train_test_split(X_experiment, y_experiment, test_size=0.2, random_state=42)

# Train a new linear regression model with the experimental features
lr_model_exp = LinearRegression()
lr_model_exp.fit(X_train_exp, y_train_exp)
y_pred_exp = lr_model_exp.predict(X_test_exp)

# Evaluate the new model
mse_exp = mean_squared_error(y_test_exp, y_pred_exp)
r2_exp = r2_score(y_test_exp, y_pred_exp)

print(f"Experimental Model MSE: {mse_exp:.2f}")
print(f"Experimental Model R-squared: {r2_exp:.2f}")
```

```
Experimental Model MSE: 6.47
Experimental Model R-squared: 0.58
```

[ ]:
```python
#Visualize Predictions
plt.figure(figsize=(8, 6))
plt.scatter(y_test_exp, y_pred_exp, color='green')
plt.plot([y_test_exp.min(), y_test_exp.max()], [y_test_exp.min(), y_test_exp.
    ↪max()], '--r')
plt.xlabel("Actual Exam Score")
plt.ylabel("Predicted Exam Score")
plt.title("Actual vs Predicted Scores (Multiple Features)")
plt.grid(True)
plt.show()
```



[ ]:
```python
```

2 Task 2: Customer Segmentation

[ ]:
```python
#libraries
import pandas as pd
import numpy as np
```

```python
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
from sklearn.cluster import DBSCAN
```

[ ]:
```python
#Data Loading
df = pd.read_csv('/content/mall_cutomer.zip')
df.head()
```

[ ]:
```
   CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)
0           1    Male   19                  15                      39
1           2    Male   21                  15                      81
2           3  Female   20                  16                       6
3           4  Female   23                  16                      77
4           5  Female   31                  17                      40
```

Data Cleaning

[ ]:
```python
print(df.info())
print(df.isnull().sum())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
None
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

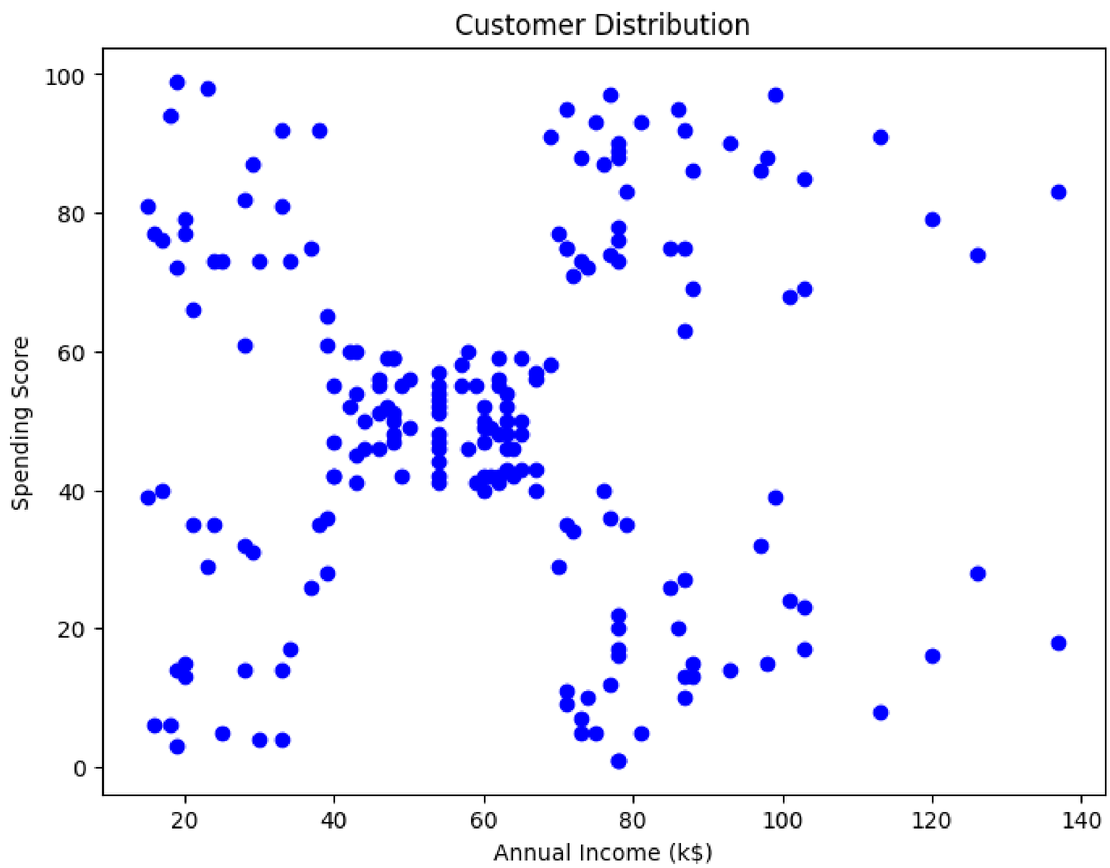
[ ]:
```python
X = df[['Annual Income (k$)', 'Spending Score (1-100)']]
```

Feature Scaling

[ ]:
```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Visualize Raw Data

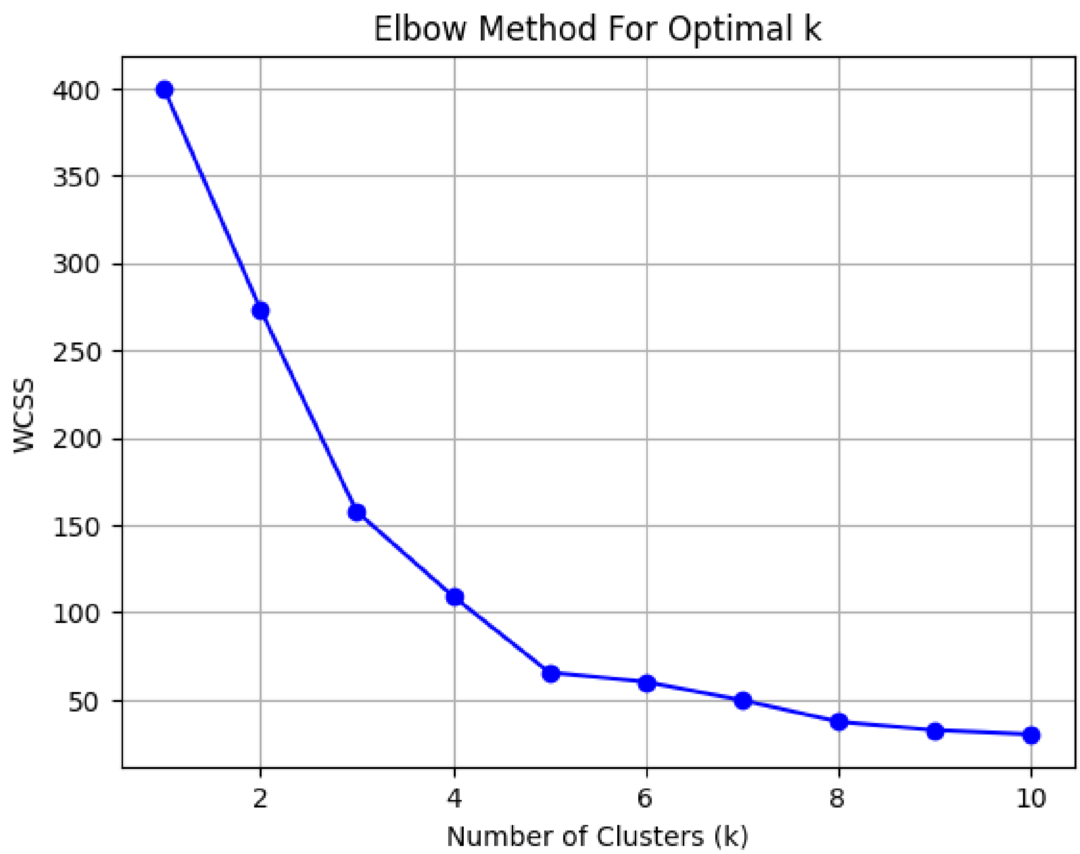
[ ]:
```
plt.figure(figsize=(8, 6))
plt.scatter(X['Annual Income (k$)'], X['Spending Score (1-100)'], c='blue')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score')
plt.title('Customer Distribution')
plt.show()
```



Optimal K using Elbow Method

[ ]:
```
wcss = []
K_range = range(1, 11)

for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X_scaled)
```

```
    wcss.append(kmeans.inertia_)

plt.plot(K_range, wcss, 'bo-')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('WCSS')
plt.title('Elbow Method For Optimal k')
plt.grid(True)
plt.show()
```



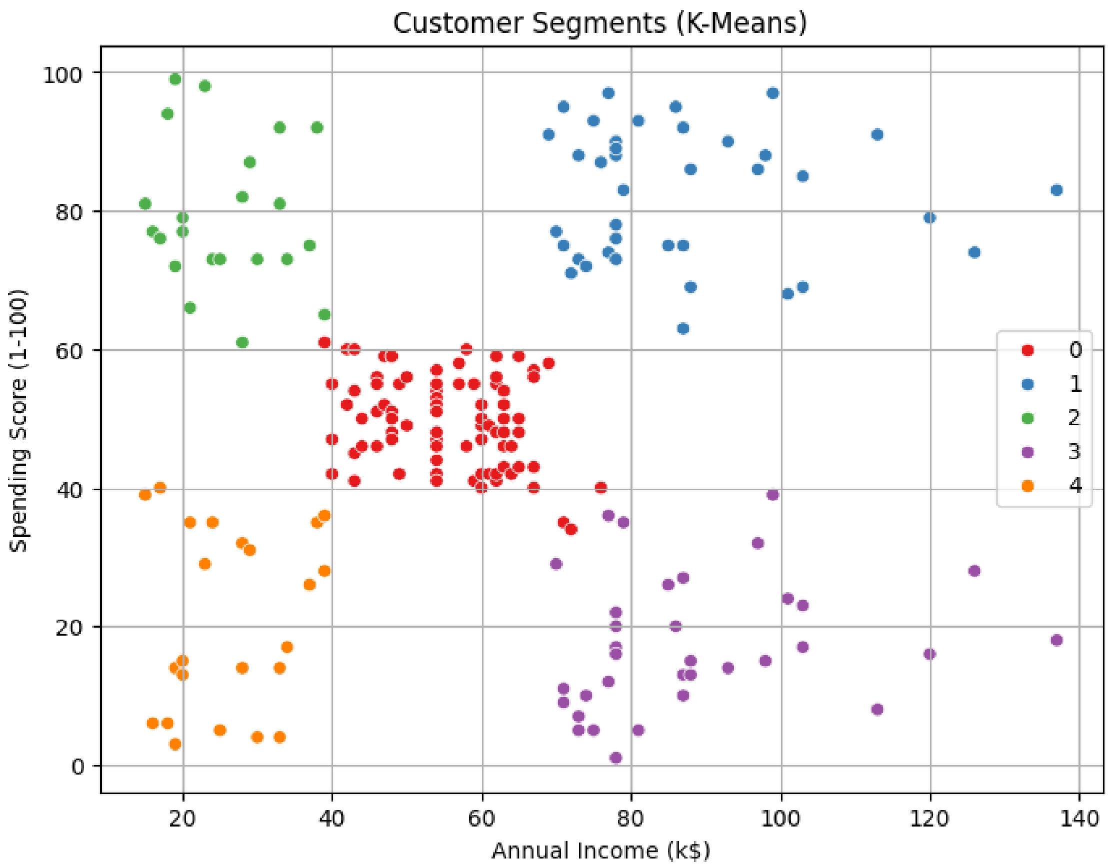
K- Means Clustering

```
[ ]: kmeans = KMeans(n_clusters=5, random_state=42)
     df['Cluster'] = kmeans.fit_predict(X_scaled)
```

Visualize Cluster

```
[ ]: plt.figure(figsize=(8, 6))
     sns.scatterplot(data=df, x='Annual Income (k$)', y='Spending Score (1-100)',␣
      ↪hue='Cluster', palette='Set1')
     plt.title('Customer Segments (K-Means)')
```

```
plt.legend()
plt.grid(True)
plt.show()
```



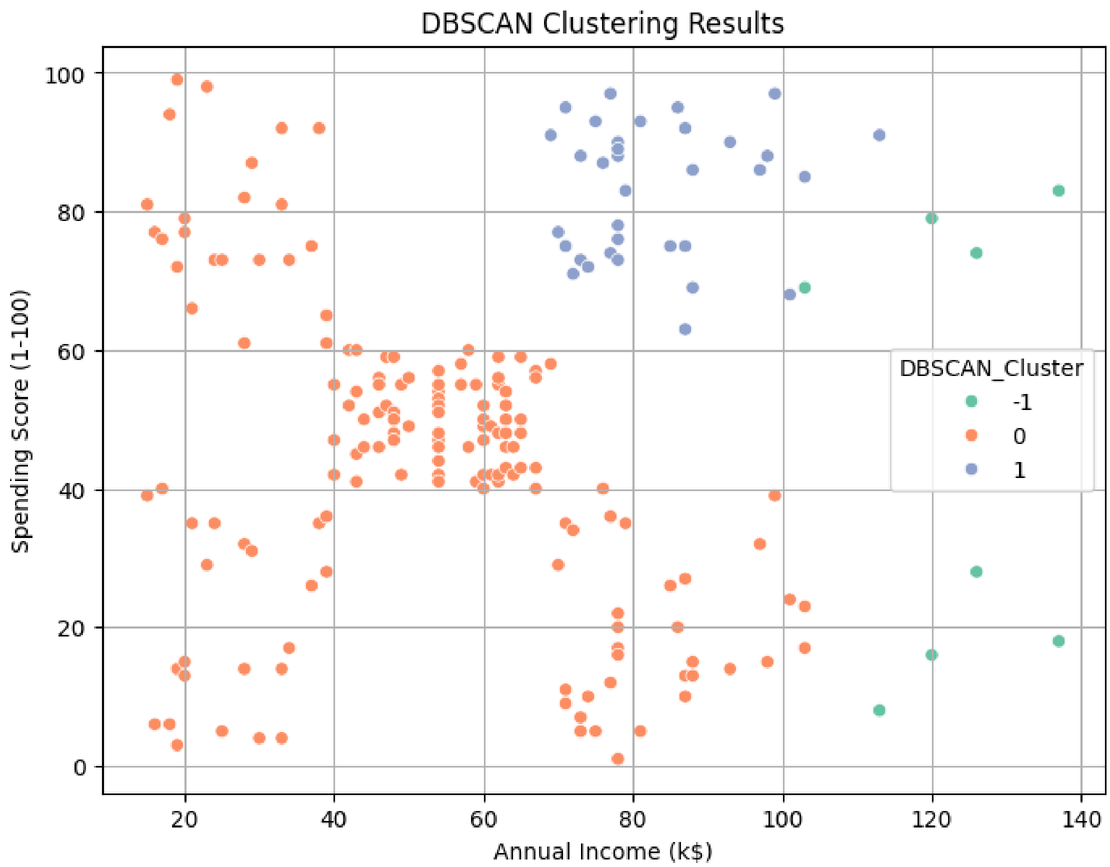
Analyze Clusters

[ ]:
```
cluster_summary = df.groupby('Cluster')[['Annual Income (k$)', 'Spending Score␣
↪(1-100)']].mean()
print("  Average values per cluster:\n")
print(cluster_summary)
```

```
  Average values per cluster:

         Annual Income (k$)  Spending Score (1-100)
Cluster
0                 55.296296               49.518519
1                 86.538462               82.128205
2                 25.727273               79.363636
3                 88.200000               17.114286
4                 26.304348               20.913043
```

Try DBSCAN

```
[ ]: dbscan = DBSCAN(eps=0.5, min_samples=5)
df['DBSCAN_Cluster'] = dbscan.fit_predict(X_scaled)

plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='Annual Income (k$)', y='Spending Score (1-100)',␣
  ↪hue='DBSCAN_Cluster', palette='Set2')
plt.title('DBSCAN Clustering Results')
plt.grid(True)
plt.show()
```



Analyze Average Spending per DBSCAN Cluster

```
[ ]: # Group by DBSCAN cluster and compute means
dbscan_summary = df.groupby('DBSCAN_Cluster')[['Annual Income (k$)', 'Spending␣
  ↪Score (1-100)']].mean().round(2)

print("  Average values per DBSCAN cluster:\n")
print(dbscan_summary)
```

```
Average values per DBSCAN cluster:

                Annual Income (k$)  Spending Score (1-100)
DBSCAN_Cluster
-1                          122.75                   46.88
 0                           52.49                   43.10
 1                           82.54                   82.80
```

[ ]:

```
```

3 Task 3: Forest Cover Type Classification

[ ]:

```python
#Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪ConfusionMatrixDisplay
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.preprocessing import StandardScaler
```

[ ]:

```python
#Load Dataset
df = pd.read_csv('/content/Forest_cover.zip')
df.head()
```

[ ]:

```
   Elevation  Aspect  Slope  Horizontal_Distance_To_Hydrology  \
0       2596      51      3                               258
1       2590      56      2                               212
2       2804     139      9                               268
3       2785     155     18                               242
4       2595      45      2                               153

   Vertical_Distance_To_Hydrology  Horizontal_Distance_To_Roadways  \
0                               0                              510
1                              -6                              390
2                              65                             3180
3                             118                             3090
4                              -1                              391

   Hillshade_9am  Hillshade_Noon  Hillshade_3pm  \
0            221             232            148
1            220             235            151
```

```
2           234           238           135
3           238           238           122
4           220           234           150

   Horizontal_Distance_To_Fire_Points  …  Soil_Type32  Soil_Type33  \
0                                6279  …            0            0
1                                6225  …            0            0
2                                6121  …            0            0
3                                6211  …            0            0
4                                6172  …            0            0

   Soil_Type34  Soil_Type35  Soil_Type36  Soil_Type37  Soil_Type38  \
0            0            0            0            0            0
1            0            0            0            0            0
2            0            0            0            0            0
3            0            0            0            0            0
4            0            0            0            0            0

   Soil_Type39  Soil_Type40  Cover_Type
0            0            0           5
1            0            0           5
2            0            0           2
3            0            0           2
4            0            0           5

[5 rows x 55 columns]
```

Explore and Data Clean

[ ]:
```
df.info()
print(df.isnull().sum())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 581012 entries, 0 to 581011
Data columns (total 55 columns):
 #   Column                              Non-Null Count   Dtype
---  ------                              --------------   -----
 0   Elevation                           581012 non-null  int64
 1   Aspect                              581012 non-null  int64
 2   Slope                               581012 non-null  int64
 3   Horizontal_Distance_To_Hydrology    581012 non-null  int64
 4   Vertical_Distance_To_Hydrology      581012 non-null  int64
 5   Horizontal_Distance_To_Roadways     581012 non-null  int64
 6   Hillshade_9am                       581012 non-null  int64
 7   Hillshade_Noon                      581012 non-null  int64
 8   Hillshade_3pm                       581012 non-null  int64
 9   Horizontal_Distance_To_Fire_Points  581012 non-null  int64
 10  Wilderness_Area1                    581012 non-null  int64
```

```
 11  Wilderness_Area2                    581012 non-null  int64
 12  Wilderness_Area3                    581012 non-null  int64
 13  Wilderness_Area4                    581012 non-null  int64
 14  Soil_Type1                          581012 non-null  int64
 15  Soil_Type2                          581012 non-null  int64
 16  Soil_Type3                          581012 non-null  int64
 17  Soil_Type4                          581012 non-null  int64
 18  Soil_Type5                          581012 non-null  int64
 19  Soil_Type6                          581012 non-null  int64
 20  Soil_Type7                          581012 non-null  int64
 21  Soil_Type8                          581012 non-null  int64
 22  Soil_Type9                          581012 non-null  int64
 23  Soil_Type10                         581012 non-null  int64
 24  Soil_Type11                         581012 non-null  int64
 25  Soil_Type12                         581012 non-null  int64
 26  Soil_Type13                         581012 non-null  int64
 27  Soil_Type14                         581012 non-null  int64
 28  Soil_Type15                         581012 non-null  int64
 29  Soil_Type16                         581012 non-null  int64
 30  Soil_Type17                         581012 non-null  int64
 31  Soil_Type18                         581012 non-null  int64
 32  Soil_Type19                         581012 non-null  int64
 33  Soil_Type20                         581012 non-null  int64
 34  Soil_Type21                         581012 non-null  int64
 35  Soil_Type22                         581012 non-null  int64
 36  Soil_Type23                         581012 non-null  int64
 37  Soil_Type24                         581012 non-null  int64
 38  Soil_Type25                         581012 non-null  int64
 39  Soil_Type26                         581012 non-null  int64
 40  Soil_Type27                         581012 non-null  int64
 41  Soil_Type28                         581012 non-null  int64
 42  Soil_Type29                         581012 non-null  int64
 43  Soil_Type30                         581012 non-null  int64
 44  Soil_Type31                         581012 non-null  int64
 45  Soil_Type32                         581012 non-null  int64
 46  Soil_Type33                         581012 non-null  int64
 47  Soil_Type34                         581012 non-null  int64
 48  Soil_Type35                         581012 non-null  int64
 49  Soil_Type36                         581012 non-null  int64
 50  Soil_Type37                         581012 non-null  int64
 51  Soil_Type38                         581012 non-null  int64
 52  Soil_Type39                         581012 non-null  int64
 53  Soil_Type40                         581012 non-null  int64
 54  Cover_Type                          581012 non-null  int64
dtypes: int64(55)
memory usage: 243.8 MB
Elevation                             0
Aspect                                0
```

Slope	0
Horizontal_Distance_To_Hydrology	0
Vertical_Distance_To_Hydrology	0
Horizontal_Distance_To_Roadways	0
Hillshade_9am	0
Hillshade_Noon	0
Hillshade_3pm	0
Horizontal_Distance_To_Fire_Points	0
Wilderness_Area1	0
Wilderness_Area2	0
Wilderness_Area3	0
Wilderness_Area4	0
Soil_Type1	0
Soil_Type2	0
Soil_Type3	0
Soil_Type4	0
Soil_Type5	0
Soil_Type6	0
Soil_Type7	0
Soil_Type8	0
Soil_Type9	0
Soil_Type10	0
Soil_Type11	0
Soil_Type12	0
Soil_Type13	0
Soil_Type14	0
Soil_Type15	0
Soil_Type16	0
Soil_Type17	0
Soil_Type18	0
Soil_Type19	0
Soil_Type20	0
Soil_Type21	0
Soil_Type22	0
Soil_Type23	0
Soil_Type24	0
Soil_Type25	0
Soil_Type26	0
Soil_Type27	0
Soil_Type28	0
Soil_Type29	0
Soil_Type30	0
Soil_Type31	0
Soil_Type32	0
Soil_Type33	0
Soil_Type34	0
Soil_Type35	0
Soil_Type36	0

```
Soil_Type37                          0
Soil_Type38                          0
Soil_Type39                          0
Soil_Type40                          0
Cover_Type                           0
dtype: int64
```

Feature and Target split

```
[ ]: X = df.drop('Cover_Type', axis=1)
     y = df['Cover_Type'] - 1 # Subtract 1 to make labels 0-indexed
```

Scale Numeric Feature

```
[ ]: scaler = StandardScaler()
     X_scaled = scaler.fit_transform(X)
```

Train Test Split

```
[ ]: X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,␣
      ↪random_state=42, stratify=y)
```

Train Random Forest Classifier

```
[ ]: rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
     rf_model.fit(X_train, y_train)
     rf_preds = rf_model.predict(X_test)
```

Train XGBoost Classifier

```
[ ]: xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='mlogloss',␣
      ↪random_state=42)
     xgb_model.fit(X_train, y_train)
     xgb_preds = xgb_model.predict(X_test)
```

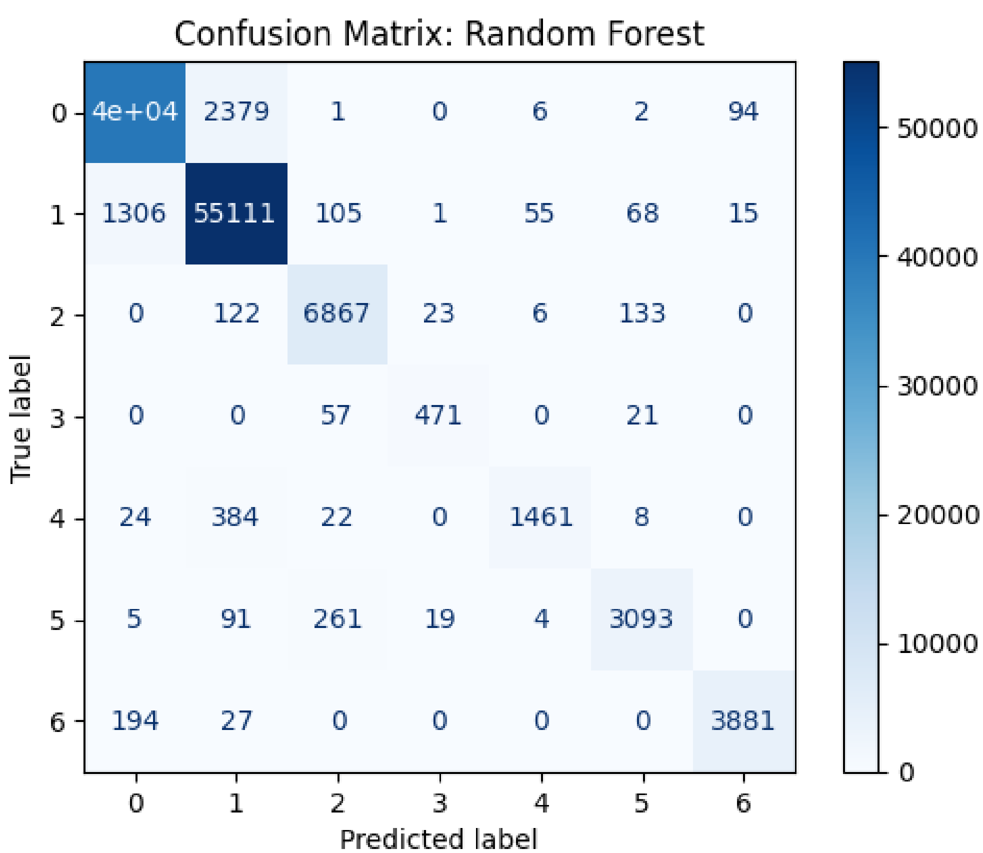
```
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning:
[13:40:53] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
```

Evaluate Models

```
[ ]: def evaluate_model(name, y_true, y_pred):
         print(f"\n  Classification Report for {name}:\n")
         print(classification_report(y_true, y_pred))
         cm = confusion_matrix(y_true, y_pred)
         disp = ConfusionMatrixDisplay(confusion_matrix=cm)
         disp.plot(cmap='Blues')
         plt.title(f'Confusion Matrix: {name}')
         plt.show()
```

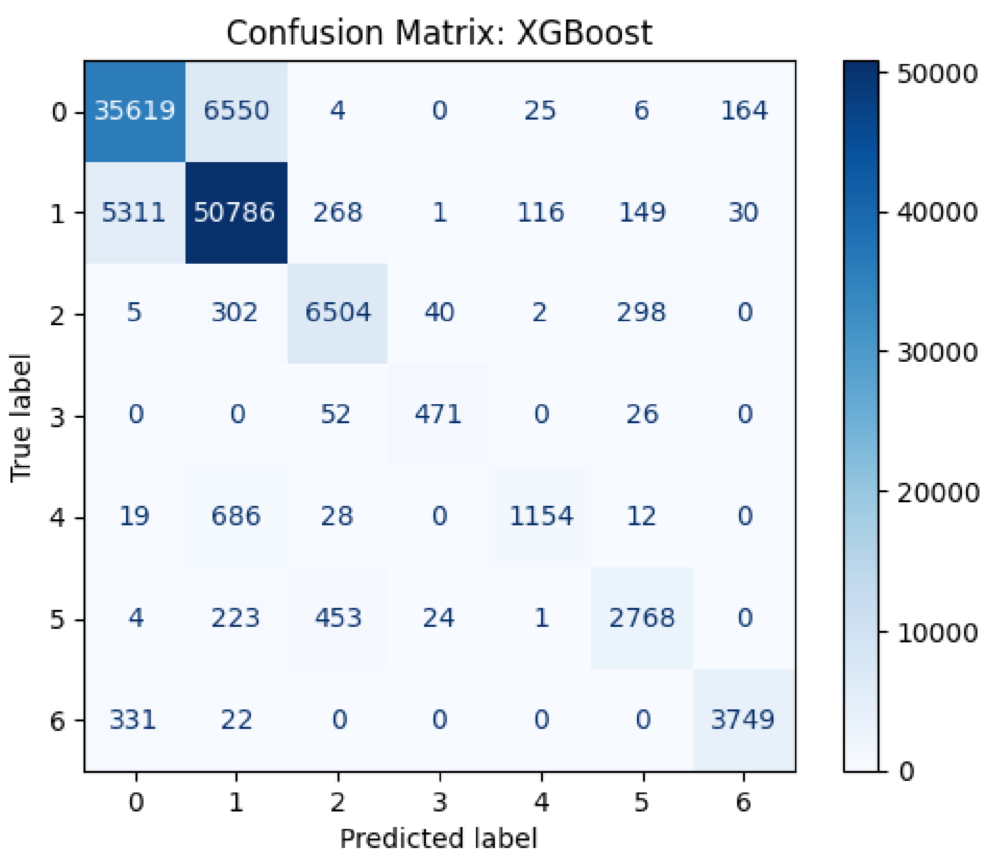
```python
evaluate_model("Random Forest", y_test, rf_preds)
evaluate_model("XGBoost", y_test, xgb_preds)
```

```
Classification Report for Random Forest:

              precision    recall  f1-score   support

           0       0.96      0.94      0.95     42368
           1       0.95      0.97      0.96     56661
           2       0.94      0.96      0.95      7151
           3       0.92      0.86      0.89       549
           4       0.95      0.77      0.85      1899
           5       0.93      0.89      0.91      3473
           6       0.97      0.95      0.96      4102

    accuracy                           0.95    116203
   macro avg       0.95      0.91      0.92    116203
weighted avg       0.95      0.95      0.95    116203
```



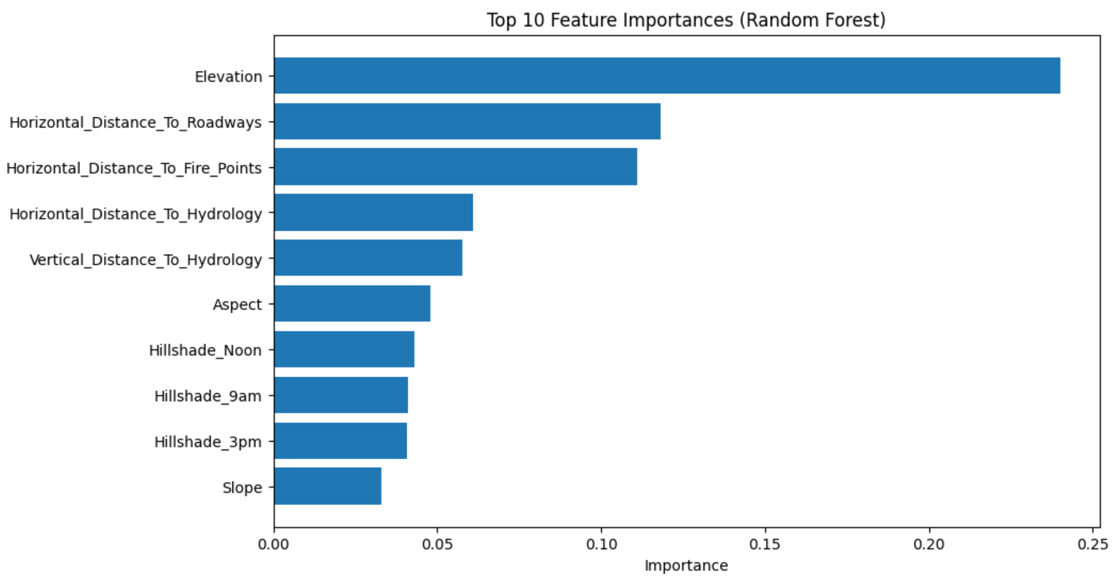
Classification Report for XGBoost:

	precision	recall	f1-score	support
0	0.86	0.84	0.85	42368
1	0.87	0.90	0.88	56661
2	0.89	0.91	0.90	7151
3	0.88	0.86	0.87	549
4	0.89	0.61	0.72	1899
5	0.85	0.80	0.82	3473
6	0.95	0.91	0.93	4102
accuracy			0.87	116203
macro avg	0.88	0.83	0.85	116203
weighted avg	0.87	0.87	0.87	116203



Feature Importance

[ ]:
```python
# Random Forest
importances_rf = rf_model.feature_importances_
indices_rf = np.argsort(importances_rf)[-10:]  # top 10

plt.figure(figsize=(10, 6))
plt.barh(range(len(indices_rf)), importances_rf[indices_rf], align='center')
plt.yticks(range(len(indices_rf)), [X.columns[i] for i in indices_rf])
plt.title("Top 10 Feature Importances (Random Forest)")
plt.xlabel("Importance")
plt.show()

# XGBoost
xgb_model.feature_importances_[:10]
```



[ ]:
```
array([0.09264691, 0.0071369 , 0.00419376, 0.01318378, 0.00703175,
        0.01357988, 0.00856525, 0.01079951, 0.00502013, 0.01228627],
       dtype=float32)
```

Compare Random Forest vs. XGBoost

[ ]:
```python
from sklearn.metrics import accuracy_score

acc_rf = accuracy_score(y_test, rf_preds)
acc_xgb = accuracy_score(y_test, xgb_preds)

print(f"  Random Forest Accuracy: {acc_rf:.4f}")
print(f"  XGBoost Accuracy: {acc_xgb:.4f}")
```

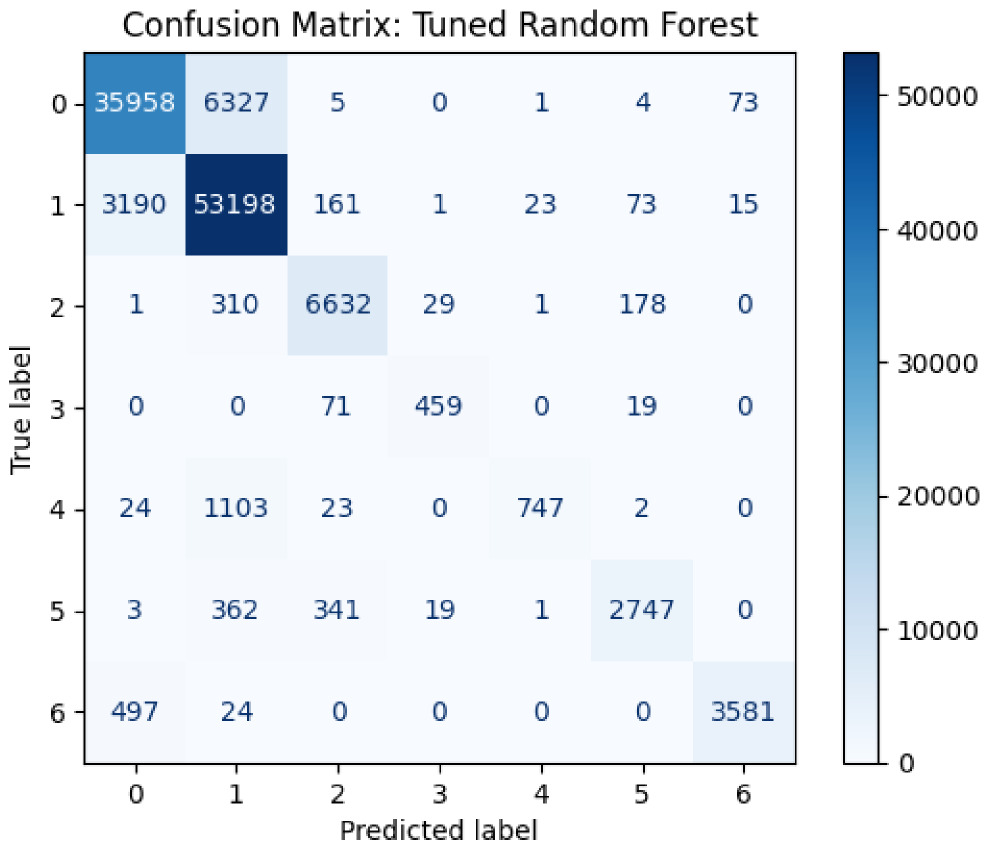
```
 Random Forest Accuracy: 0.9532
```

```
XGBoost Accuracy: 0.8696
```

Hyperparameter Tuning

```
[ ]: param_grid = {
         'n_estimators': [50, 100],
         'max_depth': [10, 20],
         'min_samples_split': [2, 5]
     }

     grid_rf = GridSearchCV(RandomForestClassifier(random_state=42), param_grid,␣
      ↪cv=3, scoring='accuracy', verbose=1)
     grid_rf.fit(X_train, y_train)

     print("  Best Random Forest Params:", grid_rf.best_params_)
     best_rf = grid_rf.best_estimator_
     evaluate_model("Tuned Random Forest", y_test, best_rf.predict(X_test))
```

```
Fitting 3 folds for each of 8 candidates, totalling 24 fits
  Best Random Forest Params: {'max_depth': 20, 'min_samples_split': 2,
'n_estimators': 100}

  Classification Report for Tuned Random Forest:

              precision    recall  f1-score   support

           0       0.91      0.85      0.88     42368
           1       0.87      0.94      0.90     56661
           2       0.92      0.93      0.92      7151
           3       0.90      0.84      0.87       549
           4       0.97      0.39      0.56      1899
           5       0.91      0.79      0.85      3473
           6       0.98      0.87      0.92      4102

    accuracy                           0.89    116203
   macro avg       0.92      0.80      0.84    116203
weighted avg       0.89      0.89      0.89    116203
```



Hyperparameter Tuning for XGBoost

```
[ ]: param_grid_xgb = {
         'n_estimators': [50, 100],
         'max_depth': [3, 6],
         'learning_rate': [0.01, 0.1]
     }

     grid_xgb = GridSearchCV(XGBClassifier(use_label_encoder=False,␣
      ↪eval_metric='mlogloss', random_state=42), param_grid_xgb, cv=3,␣
      ↪scoring='accuracy', verbose=1)
     grid_xgb.fit(X_train, y_train)

     print("  Best XGBoost Params:", grid_xgb.best_params_)
     best_xgb = grid_xgb.best_estimator_
     evaluate_model("Tuned XGBoost", y_test, best_xgb.predict(X_test))
```

```
Fitting 3 folds for each of 8 candidates, totalling 24 fits
```

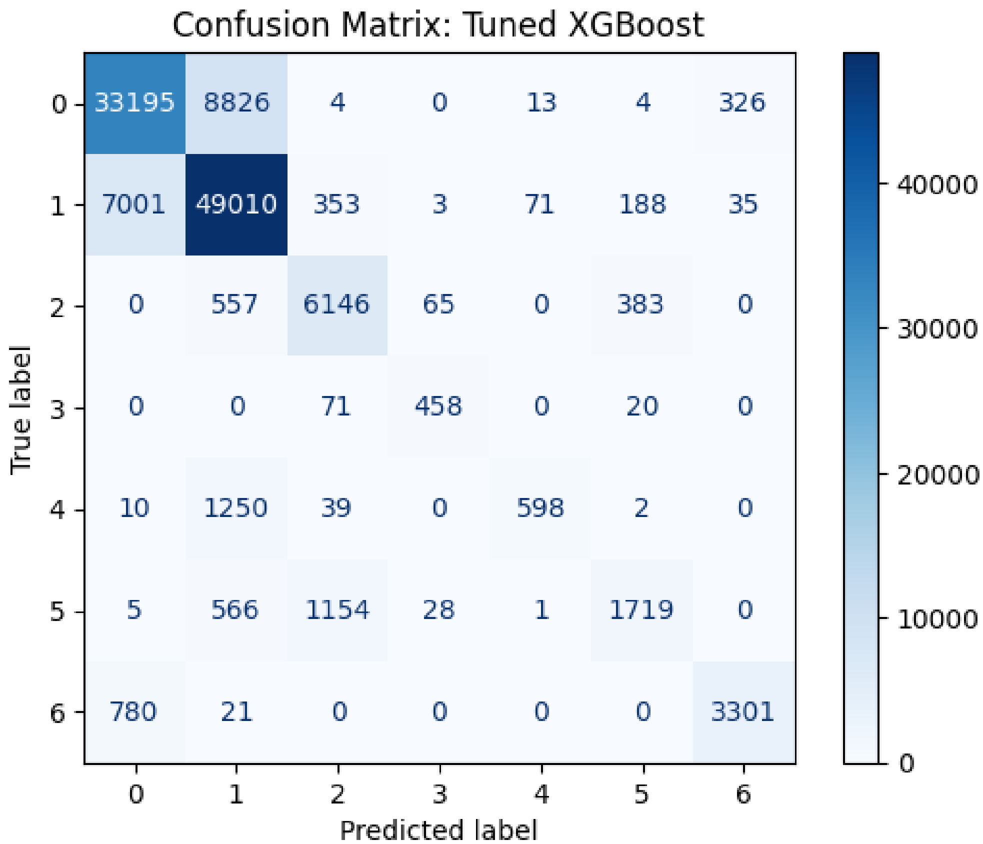
```
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning:
[14:04:00] WARNING: /workspace/src/learner.cc:738:
```

```
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning:
[14:04:13] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning:
[14:04:25] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning:
[14:04:38] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning:
[14:05:02] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning:
[14:05:28] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning:
[14:05:52] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning:
[14:06:09] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning:
[14:06:26] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning:
[14:06:43] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
```

```
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning:
[14:07:16] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning:
[14:07:48] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning:
[14:08:20] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning:
[14:08:34] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning:
[14:08:47] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning:
[14:09:00] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning:
[14:09:28] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning:
[14:09:53] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning:
[14:10:19] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning:
[14:10:35] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.
```

```
  bst.update(dtrain, iteration=i, fobj=obj)
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning:
[14:10:52] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning:
[14:11:09] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning:
[14:11:40] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning:
[14:12:13] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
/usr/local/lib/python3.11/dist-packages/xgboost/training.py:183: UserWarning:
[14:12:46] WARNING: /workspace/src/learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
```

```
 Best XGBoost Params: {'learning_rate': 0.1, 'max_depth': 6, 'n_estimators':
100}

 Classification Report for Tuned XGBoost:

              precision    recall  f1-score   support

           0       0.81      0.78      0.80     42368
           1       0.81      0.86      0.84     56661
           2       0.79      0.86      0.82      7151
           3       0.83      0.83      0.83       549
           4       0.88      0.31      0.46      1899
           5       0.74      0.49      0.59      3473
           6       0.90      0.80      0.85      4102

    accuracy                           0.81    116203
   macro avg       0.82      0.71      0.74    116203
weighted avg       0.81      0.81      0.81    116203
```



4 Task 4: Loan Approval Prediction Description

[ ]:
```
#Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import classification_report, confusion_matrix,␣
↪ConfusionMatrixDisplay
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from imblearn.over_sampling import SMOTE
```

[ ]:
```
#Load Dataset
import zipfile
```

```
with zipfile.ZipFile('/content/loan_approval.zip', 'r') as zip_ref:
    zip_ref.extractall('/content/')

df_train = pd.read_csv('/content/train_u6lujuX_CVtuZ9i.csv')

print("Training data:")
display(df_train.head())
```

```
Training data:

    Loan_ID Gender Married Dependents     Education Self_Employed  \
0  LP001002   Male      No          0      Graduate            No
1  LP001003   Male     Yes          1      Graduate            No
2  LP001005   Male     Yes          0      Graduate           Yes
3  LP001006   Male     Yes          0  Not Graduate            No
4  LP001008   Male      No          0      Graduate            No

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0             5849                0.0         NaN             360.0
1             4583             1508.0       128.0             360.0
2             3000                0.0        66.0             360.0
3             2583             2358.0       120.0             360.0
4             6000                0.0       141.0             360.0

   Credit_History Property_Area Loan_Status
0             1.0         Urban           Y
1             1.0         Rural           N
2             1.0         Urban           Y
3             1.0         Urban           Y
4             1.0         Urban           Y
```

Exploratory Data Analysis(EDA)

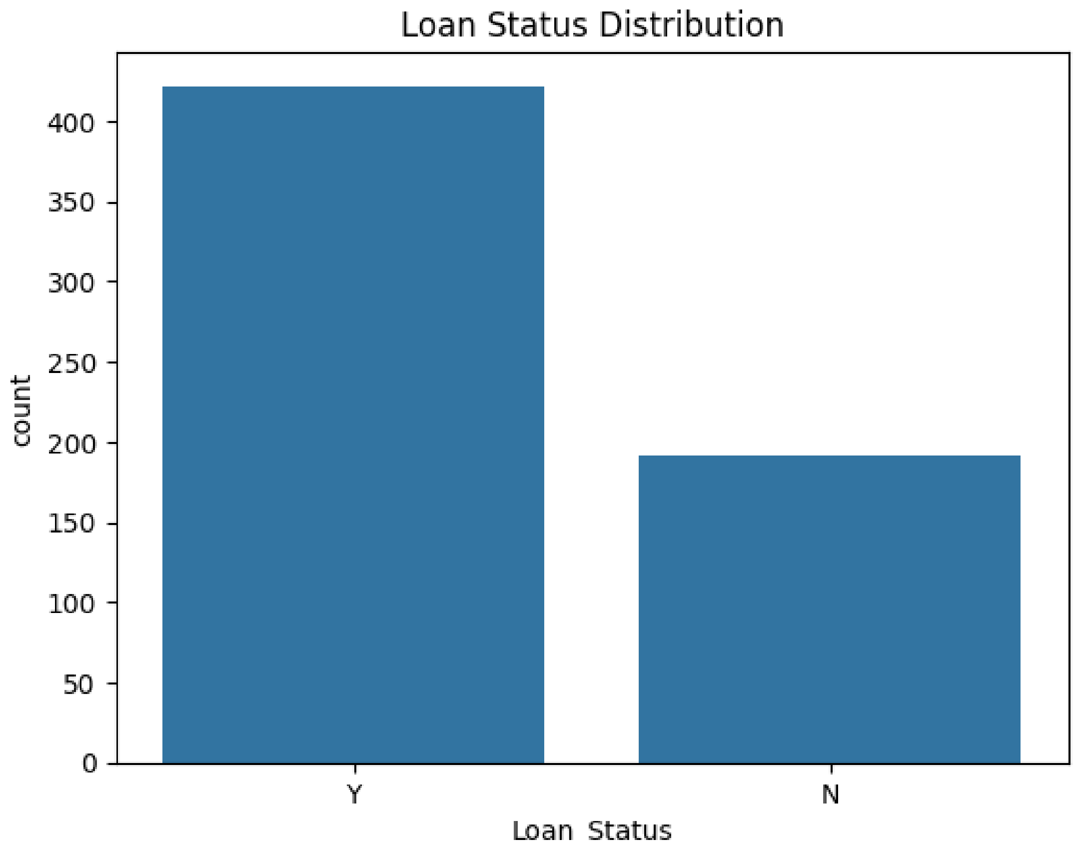
[ ]:
```
print(df_train.info())
print(df_train.isnull().sum())
print(df_train['Loan_Status'].value_counts())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
```

```
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
None
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
Loan_Status
Y    422
N    192
Name: count, dtype: int64
```

[ ]:
```python
#Visualize Class Imbalance
sns.countplot(data=df_train, x='Loan_Status')
plt.title("Loan Status Distribution")
```

[ ]:
```
Text(0.5, 1.0, 'Loan Status Distribution')
```



Handle Missing values

```
[ ]: # Fill numerical with median, categorical with mode
df_train['Gender'] = df_train['Gender'].fillna(df_train['Gender'].mode()[0])
df_train['Married'] = df_train['Married'].fillna(df_train['Married'].mode()[0])
df_train['Self_Employed'] = df_train['Self_Employed'].
  ↪fillna(df_train['Self_Employed'].mode()[0])
df_train['Credit_History'] = df_train['Credit_History'].
  ↪fillna(df_train['Credit_History'].mode()[0])
df_train['Loan_Amount_Term'] = df_train['Loan_Amount_Term'].
  ↪fillna(df_train['Loan_Amount_Term'].mode()[0])
df_train['LoanAmount'] = df_train['LoanAmount'].fillna(df_train['LoanAmount'].
  ↪median())

# Handle Dependents missing values with mode
df_train['Dependents'] = df_train['Dependents'].astype(str).
  ↪fillna(df_train['Dependents'].mode()[0])
```

Encode Categorical Varaiables

```
[ ]: le = LabelEncoder()

for col in ['Gender', 'Married', 'Education', 'Self_Employed', 'Property_Area',␣
 ↪'Dependents']:
    # Combine train and test data for fitting the encoder to handle unseen␣
 ↪labels
    combined_data = pd.concat([df_train[col], df_test[col]], axis=0).
 ↪astype(str).unique()
    le.fit(combined_data)
    df_train[col] = le.transform(df_train[col].astype(str))
    df_test[col] = le.transform(df_test[col].astype(str))

# Encode Loan_Status in train data separately as it's the target variable
df_train['Loan_Status'] = le.fit_transform(df_train['Loan_Status'])
```

Feature and Target Split

```
[ ]: X = df_train.drop(columns=['Loan_ID', 'Loan_Status'])
y = df_train['Loan_Status']
```

Train Test Split

```
[ ]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42, stratify=y)
```

Handle Imbalanced Data using SMOTE

```
[ ]: smote = SMOTE(random_state=42)
X_train_bal, y_train_bal = smote.fit_resample(X_train, y_train)

print("Class Distribution After SMOTE:\n", pd.Series(y_train_bal).
 ↪value_counts())
```

```
Class Distribution After SMOTE:
 Loan_Status
1    337
0    337
Name: count, dtype: int64
```

Train Classification Model

Logistic Regression

```
[ ]: lr = LogisticRegression(max_iter=1000)
lr.fit(X_train_bal, y_train_bal)
lr_preds = lr.predict(X_test)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:465:
ConvergenceWarning: lbfgs failed to converge (status=1):
```

```
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

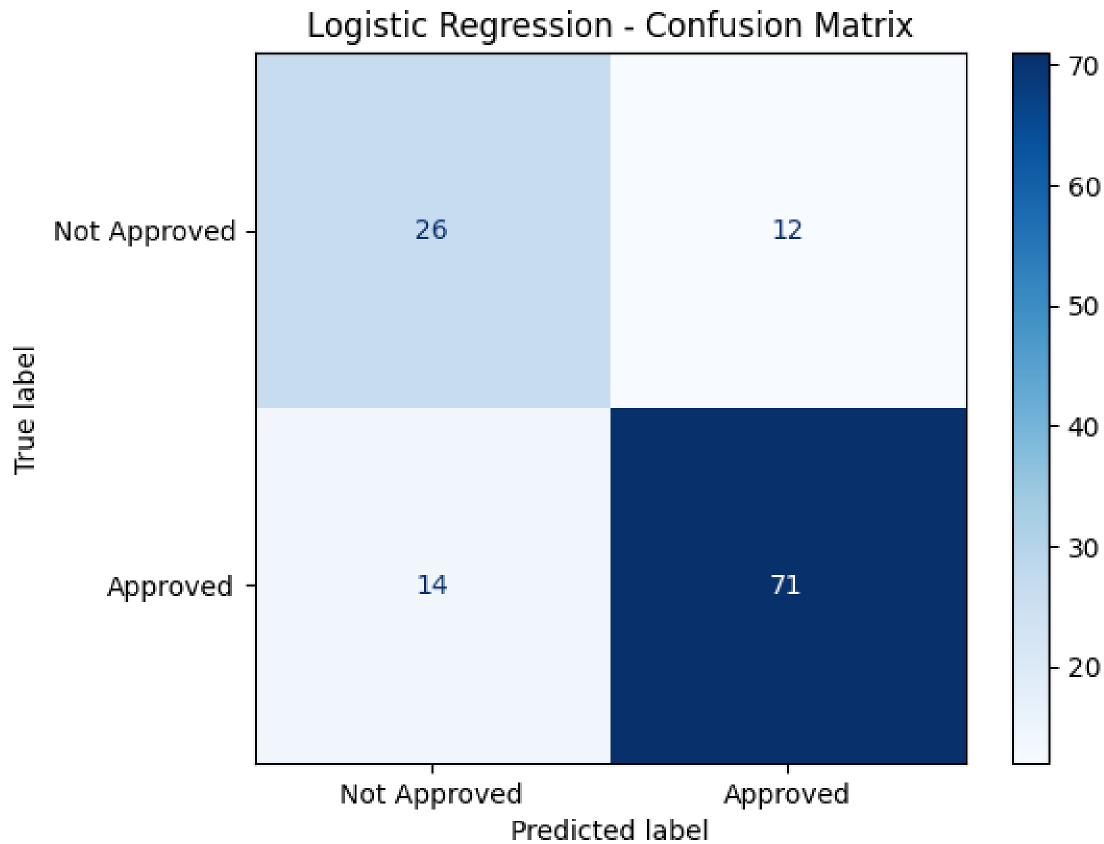
Decision Tree

```
[ ]: dt = DecisionTreeClassifier(random_state=42)
     dt.fit(X_train_bal, y_train_bal)
     dt_preds = dt.predict(X_test)
```

Evaluate Model Performance

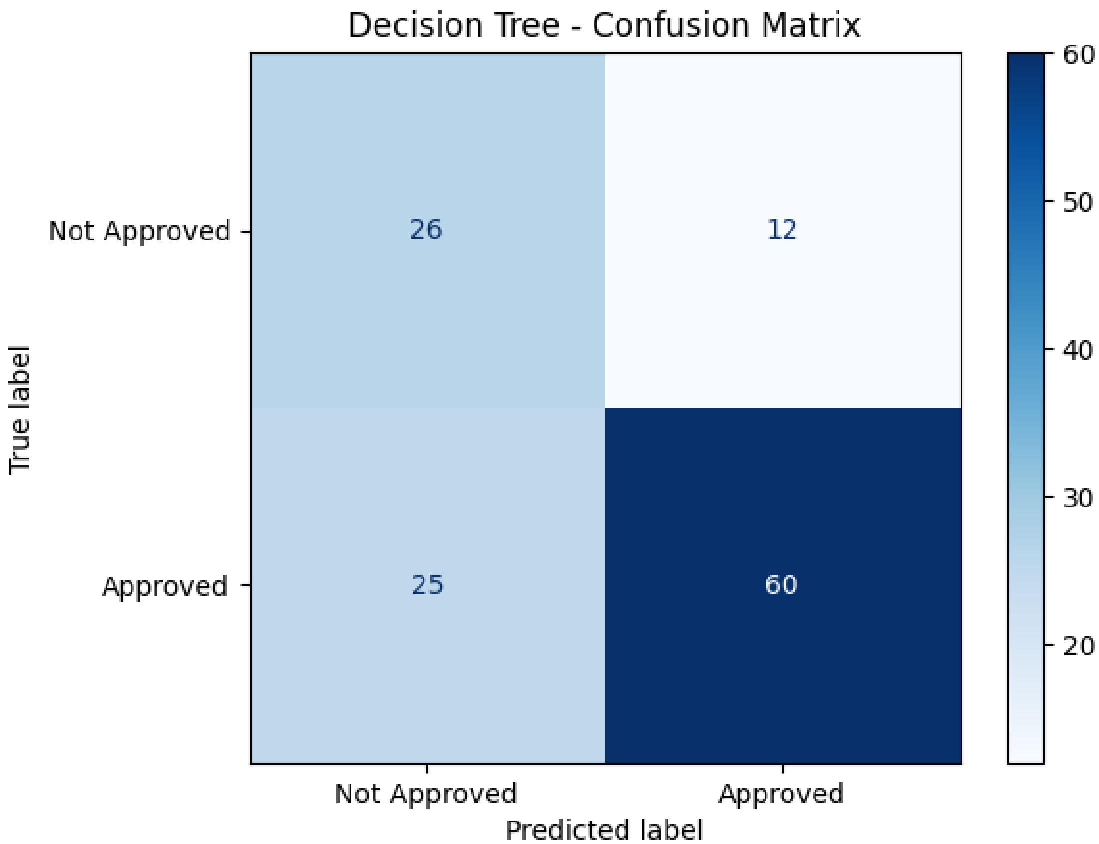
```
[ ]: def evaluate_model(name, y_true, y_pred):
         print(f"\n {name} Classification Report:")
         print(classification_report(y_true, y_pred, target_names=['Not Approved',␣
      ↪'Approved']))

         cm = confusion_matrix(y_true, y_pred)
         disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Not␣
      ↪Approved', 'Approved'])
         disp.plot(cmap='Blues')
         plt.title(f'{name} - Confusion Matrix')
         plt.show()

     evaluate_model("Logistic Regression", y_test, lr_preds)
     evaluate_model("Decision Tree", y_test, dt_preds)
```

```
 Logistic Regression Classification Report:
              precision    recall  f1-score   support

Not Approved       0.65      0.68      0.67        38
    Approved       0.86      0.84      0.85        85

    accuracy                           0.79       123
   macro avg       0.75      0.76      0.76       123
weighted avg       0.79      0.79      0.79       123
```



Decision Tree Classification Report:

	precision	recall	f1-score	support
Not Approved	0.51	0.68	0.58	38
Approved	0.83	0.71	0.76	85
accuracy			0.70	123
macro avg	0.67	0.70	0.67	123
weighted avg	0.73	0.70	0.71	123



5 Task 5: Movie Recommendation System Description

```
[ ]: #Libraries
import pandas as pd
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics import precision_score
from sklearn.metrics import pairwise_distances
from sklearn.decomposition import TruncatedSVD
```

Load and Merge Data

```
[ ]: #Load Dataset
import zipfile

# Extract the zip file
with zipfile.ZipFile('/content/Movie.zip', 'r') as zip_ref:
    zip_ref.extractall('/content/')
```

```python
# Load ratings data
ratings_columns = ['user_id', 'item_id', 'rating', 'timestamp']
ratings_df = pd.read_csv('/content/ml-100k/u.data', sep='\t',␣
 ↪names=ratings_columns)

# Load movie titles data
movies_columns = ['item_id', 'title', 'release_date', 'video_release_date',␣
 ↪'IMDb_URL', 'unknown', 'Action', 'Adventure', 'Animation', "Children's",␣
 ↪'Comedy', 'Crime', 'Documentary', 'Drama', 'Fantasy', 'Film-Noir', 'Horror',␣
 ↪'Musical', 'Mystery', 'Romance', 'Sci-Fi', 'Thriller', 'War', 'Western']
movies_df = pd.read_csv('/content/ml-100k/u.item', sep='|',␣
 ↪names=movies_columns, encoding='latin-1')

print("Ratings data:")
display(ratings_df.head())

print("\nMovie titles data:")
display(movies_df.head())
```

```
Ratings data:

   user_id  item_id  rating  timestamp
0      196      242       3  881250949
1      186      302       3  891717742
2       22      377       1  878887116
3      244       51       2  880606923
4      166      346       1  886397596


Movie titles data:

   item_id              title release_date  video_release_date  \
0        1   Toy Story (1995)  01-Jan-1995                 NaN
1        2   GoldenEye (1995)  01-Jan-1995                 NaN
2        3  Four Rooms (1995)  01-Jan-1995                 NaN
3        4  Get Shorty (1995)  01-Jan-1995                 NaN
4        5     Copycat (1995)  01-Jan-1995                 NaN

                                            IMDb_URL  unknown  Action  \
0  http://us.imdb.com/M/title-exact?Toy%20Story%2…        0       0
1  http://us.imdb.com/M/title-exact?GoldenEye%20(…        0       1
2  http://us.imdb.com/M/title-exact?Four%20Rooms%…        0       0
3  http://us.imdb.com/M/title-exact?Get%20Shorty%…        0       1
4     http://us.imdb.com/M/title-exact?Copycat%20(1995)        0       0

   Adventure  Animation  Children's  …  Fantasy  Film-Noir  Horror  Musical  \
0          0          1           1  …        0          0       0        0
1          1          0           0  …        0          0       0        0
2          0          0           0  …        0          0       0        0
```

```
3          0          0          0  …          0          0          0          0
4          0          0          0  …          0          0          0          0

   Mystery  Romance  Sci-Fi  Thriller  War  Western
0        0        0       0         0    0        0
1        0        0       0         1    0        0
2        0        0       0         1    0        0
3        0        0       0         0    0        0
4        0        0       0         1    0        0

[5 rows x 24 columns]
```

Create a User-Item Matrix

```
[ ]: # Merge ratings and movies dataframes to get movie titles
merged_df = pd.merge(ratings_df, movies_df[['item_id', 'title']], on='item_id')

# Create a User-Item Matrix
user_item_matrix = merged_df.pivot_table(index='user_id', columns='title',␣
 ↪values='rating')
user_item_matrix.fillna(0, inplace=True)
display(user_item_matrix.head())
```

```
title    'Til There Was You (1997)  1-900 (1994)  101 Dalmatians (1996)  \
user_id
1                              0.0           0.0                    2.0
2                              0.0           0.0                    0.0
3                              0.0           0.0                    0.0
4                              0.0           0.0                    0.0
5                              0.0           0.0                    2.0

title    12 Angry Men (1957)  187 (1997)  2 Days in the Valley (1996)  \
user_id
1                        5.0         0.0                          0.0
2                        0.0         0.0                          0.0
3                        0.0         2.0                          0.0
4                        0.0         0.0                          0.0
5                        0.0         0.0                          0.0

title    20,000 Leagues Under the Sea (1954)  2001: A Space Odyssey (1968)  \
user_id
1                                        3.0                           4.0
2                                        0.0                           0.0
3                                        0.0                           0.0
4                                        0.0                           0.0
5                                        0.0                           4.0

title    3 Ninjas: High Noon At Mega Mountain (1998)  39 Steps, The (1935)  \
```

```
user_id
1                                                0.0                  0.0
2                                                1.0                  0.0
3                                                0.0                  0.0
4                                                0.0                  0.0
5                                                0.0                  0.0

title     …  Yankee Zulu (1994)  Year of the Horse (1997)  \
user_id   …
1         …                 0.0                       0.0
2         …                 0.0                       0.0
3         …                 0.0                       0.0
4         …                 0.0                       0.0
5         …                 0.0                       0.0

title     You So Crazy (1994)  Young Frankenstein (1974)  Young Guns (1988)  \
user_id
1                        0.0                        5.0                3.0
2                        0.0                        0.0                0.0
3                        0.0                        0.0                0.0
4                        0.0                        0.0                0.0
5                        0.0                        4.0                0.0

title     Young Guns II (1990)  Young Poisoner's Handbook, The (1995)  \
user_id
1                          0.0                                    0.0
2                          0.0                                    0.0
3                          0.0                                    0.0
4                          0.0                                    0.0
5                          0.0                                    0.0

title     Zeus and Roxanne (1997)  unknown  Á köldum klaka (Cold Fever) (1994)
user_id
1                             0.0      4.0                                 0.0
2                             0.0      0.0                                 0.0
3                             0.0      0.0                                 0.0
4                             0.0      0.0                                 0.0
5                             0.0      4.0                                 0.0

[5 rows x 1664 columns]
```

Compute User Similarity (Cosine)

[ ]:
```python
from sklearn.metrics.pairwise import cosine_similarity

# Compute cosine similarity between users
user_similarity = cosine_similarity(user_item_matrix)
```

```
user_similarity_df = pd.DataFrame(user_similarity, index=user_item_matrix.
↪index, columns=user_item_matrix.index)

user_similarity_df.head()
```

```
[ ]: user_id        1         2         3         4         5         6         7    \
user_id
1        1.000000  0.168937  0.048388  0.064561  0.379670  0.429682  0.443097
2        0.168937  1.000000  0.113393  0.179694  0.073623  0.242106  0.108604
3        0.048388  0.113393  1.000000  0.349781  0.021592  0.074018  0.067423
4        0.064561  0.179694  0.349781  1.000000  0.031804  0.068431  0.091507
5        0.379670  0.073623  0.021592  0.031804  1.000000  0.238636  0.374733

user_id        8         9         10   …       934       935       936  \
user_id                                 …
1        0.320079  0.078385  0.377733  …  0.372213  0.119860  0.269860
2        0.104257  0.162470  0.161273  …  0.147095  0.310661  0.363328
3        0.084419  0.062039  0.066217  …  0.033885  0.043453  0.167140
4        0.188060  0.101284  0.060859  …  0.054615  0.036784  0.133619
5        0.248930  0.056847  0.201427  …  0.340183  0.080580  0.095284

user_id       937       938       939       940       941       942       943
user_id
1        0.193343  0.197949  0.118722  0.315064  0.149086  0.181612  0.399432
2        0.410725  0.322713  0.231096  0.228793  0.162911  0.175273  0.106732
3        0.071288  0.126278  0.026758  0.164539  0.102899  0.136757  0.026990
4        0.196561  0.146058  0.030202  0.196858  0.152041  0.171538  0.058752
5        0.081053  0.148607  0.071612  0.239955  0.139595  0.153799  0.313941

[5 rows x 943 columns]
```

Recommend Movies for a Target User

```
[ ]: def recommend_movies(user_id, user_item_matrix, similarity_matrix,␣
↪n_recommendations=5):
    similar_users = similarity_matrix[user_id].sort_values(ascending=False)[1:
↪11]  # Top 10 similar users
    weighted_ratings = pd.Series(dtype=float)

    for sim_user, similarity_score in similar_users.items():
        sim_user_ratings = user_item_matrix.loc[sim_user]
        weighted_ratings = weighted_ratings.add(sim_user_ratings *␣
↪similarity_score, fill_value=0)

    # Remove movies already rated by the user
    seen_movies = user_item_matrix.loc[user_id][user_item_matrix.loc[user_id] >␣
↪0].index
```

```
    weighted_ratings = weighted_ratings.drop(seen_movies, errors='ignore')

    return weighted_ratings.sort_values(ascending=False).head(n_recommendations)
```

[ ]:
```
#Recommend for user id 50
recommendations = recommend_movies(50, user_item_matrix, user_similarity_df)
print("Top recommended movies for user 50:")
print(recommendations)
```

```
Top recommended movies for user 50:
title
Star Wars (1977)                   9.834995
Welcome to the Dollhouse (1995)    8.124998
Postino, Il (1994)                 7.531862
Bound (1996)                       7.506599
Godfather, The (1972)              7.344310
dtype: float64
```

Evaluation

[ ]:
```
def precision_at_k(user_id, recommended_movies, user_item_matrix, k=5):
    actual_rated = user_item_matrix.loc[user_id]
    relevant_items = actual_rated[actual_rated >= 4].index  # consider rating␣
↪>= 4 as relevant
    recommended_set = set(recommended_movies.index[:k])
    relevant_set = set(relevant_items)

    true_positives = recommended_set.intersection(relevant_set)
    return len(true_positives) / k
```

Item-Based Collaborative Filtering

[ ]:
```
# Transpose the user-item matrix
item_user_matrix = user_item_matrix.T

# Compute item similarity
item_similarity = cosine_similarity(item_user_matrix)
item_similarity_df = pd.DataFrame(item_similarity, index=item_user_matrix.
↪index, columns=item_user_matrix.index)

item_similarity_df.head()
```

[ ]:
```
title                      'Til There Was You (1997)  1-900 (1994)  \
title
'Til There Was You (1997)                   1.000000      0.000000
1-900 (1994)                                0.000000      1.000000
101 Dalmatians (1996)                       0.024561      0.014139
12 Angry Men (1957)                         0.099561      0.009294
```

```
187 (1997)                                0.185236     0.007354

title                       101 Dalmatians (1996)  12 Angry Men (1957)  \
title
'Til There Was You (1997)               0.024561             0.099561
1-900 (1994)                            0.014139             0.009294
101 Dalmatians (1996)                   1.000000             0.167006
12 Angry Men (1957)                     0.167006             1.000000
187 (1997)                              0.061105             0.056822

title                       187 (1997)  2 Days in the Valley (1996)  \
title
'Til There Was You (1997)     0.185236                     0.159265
1-900 (1994)                  0.007354                     0.004702
101 Dalmatians (1996)         0.061105                     0.143878
12 Angry Men (1957)           0.056822                     0.167235
187 (1997)                    1.000000                     0.132327

title                       20,000 Leagues Under the Sea (1954)  \
title
'Til There Was You (1997)                              0.000000
1-900 (1994)                                           0.010055
101 Dalmatians (1996)                                  0.203781
12 Angry Men (1957)                                    0.304078
187 (1997)                                             0.042928

title                       2001: A Space Odyssey (1968)  \
title
'Til There Was You (1997)                       0.052203
1-900 (1994)                                    0.067038
101 Dalmatians (1996)                           0.225803
12 Angry Men (1957)                             0.422506
187 (1997)                                      0.065060

title                       3 Ninjas: High Noon At Mega Mountain (1998)  \
title
'Til There Was You (1997)                                      0.000000
1-900 (1994)                                                   0.000000
101 Dalmatians (1996)                                          0.027642
12 Angry Men (1957)                                            0.072682
187 (1997)                                                     0.043133

title                       39 Steps, The (1935)  …  Yankee Zulu (1994)  \
title                                             …
'Til There Was You (1997)               0.033326  …            0.000000
1-900 (1994)                            0.000000  …            0.152499
101 Dalmatians (1996)                   0.092337  …            0.000000
```

```
12 Angry Men (1957)                   0.394854  …            0.060946
187 (1997)                            0.027300  …            0.000000

title                       Year of the Horse (1997)  You So Crazy (1994)  \
title
'Til There Was You (1997)                   0.000000             0.000000
1-900 (1994)                                0.015484             0.000000
101 Dalmatians (1996)                       0.021965             0.030905
12 Angry Men (1957)                         0.016502             0.000000
187 (1997)                                  0.141997             0.000000

title                       Young Frankenstein (1974)  Young Guns (1988)  \
title
'Til There Was You (1997)                    0.027774           0.118840
1-900 (1994)                                 0.069284           0.018243
101 Dalmatians (1996)                        0.274877           0.204267
12 Angry Men (1957)                          0.403270           0.259436
187 (1997)                                   0.068257           0.067786

title                       Young Guns II (1990)  \
title
'Til There Was You (1997)               0.142315
1-900 (1994)                            0.023408
101 Dalmatians (1996)                   0.101199
12 Angry Men (1957)                     0.145519
187 (1997)                              0.091293

title                       Young Poisoner's Handbook, The (1995)  \
title
'Til There Was You (1997)                                0.029070
1-900 (1994)                                             0.006694
101 Dalmatians (1996)                                    0.056976
12 Angry Men (1957)                                      0.105226
187 (1997)                                               0.099490

title                       Zeus and Roxanne (1997)   unknown  \
title
'Til There Was You (1997)                  0.000000  0.110208
1-900 (1994)                               0.079640  0.042295
101 Dalmatians (1996)                      0.172155  0.045714
12 Angry Men (1957)                        0.038901  0.060101
187 (1997)                                 0.025184  0.142667

title                       Á köldum klaka (Cold Fever) (1994)
title
'Til There Was You (1997)                             0.000000
1-900 (1994)                                          0.000000
```

```
101 Dalmatians (1996)                                      0.000000
12 Angry Men (1957)                                        0.081261
187 (1997)                                                 0.096449

[5 rows x 1664 columns]
```

[ ]:
```python
#To recommend based on similar items:
def recommend_similar_items(movie_name, item_user_matrix, item_similarity_df,␣
 ↪n=5):
    similar_scores = item_similarity_df[movie_name].
 ↪sort_values(ascending=False)[1:n+1]
    return similar_scores
```

[ ]:
```python
#Example
print("Movies similar to 'Star Wars (1977)':")
print(recommend_similar_items("Star Wars (1977)", item_user_matrix,␣
 ↪item_similarity_df))
```

```
Movies similar to 'Star Wars (1977)':
title
Return of the Jedi (1983)          0.884476
Raiders of the Lost Ark (1981)     0.764885
Empire Strikes Back, The (1980)    0.749819
Toy Story (1995)                   0.734572
Godfather, The (1972)              0.697332
Name: Star Wars (1977), dtype: float64
```

Matrix Factorization using SVD

[ ]:
```python
from scipy.sparse.linalg import svds
import numpy as np

# Convert to numpy matrix
R = user_item_matrix.values
user_ratings_mean = np.mean(R, axis=1)
R_demeaned = R - user_ratings_mean.reshape(-1, 1)

# Perform SVD
U, sigma, Vt = svds(R_demeaned, k=50)
sigma = np.diag(sigma)

# Reconstruct the predicted ratings
predicted_ratings = np.dot(np.dot(U, sigma), Vt) + user_ratings_mean.
 ↪reshape(-1, 1)
predicted_df = pd.DataFrame(predicted_ratings, columns=user_item_matrix.
 ↪columns, index=user_item_matrix.index)
```

```
[ ]: #Recommend for User 50 using SVD
def recommend_from_svd(user_id, predictions_df, user_item_matrix, n=5):
    user_row = predictions_df.loc[user_id]
    watched_movies = user_item_matrix.loc[user_id][user_item_matrix.
 ↪loc[user_id] > 0].index
    recommendations = user_row.drop(watched_movies, errors='ignore').
 ↪sort_values(ascending=False).head(n)
    return recommendations

print("SVD Recommendations for User 50:")
print(recommend_from_svd(50, predicted_df, user_item_matrix))
```

```
SVD Recommendations for User 50:
title
Bound (1996)                         1.306506
Twelve Monkeys (1995)                1.252929
Boogie Nights (1997)                 1.173483
Welcome to the Dollhouse (1995)      1.139967
Game, The (1997)                     0.979868
Name: 50, dtype: float64
```

```
[ ]:
```

Task 6: Music Genre Classification Description

```
[ ]: import zipfile
import os

zip_path = "/content/drive/MyDrive/music genre.zip"
extract_path = "/content/music_genre"

with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_path)

# Check extracted files
for root, dirs, files in os.walk(extract_path):
    print(root, len(files))
```

```
/content/music_genre 0
/content/music_genre/Data 2
/content/music_genre/Data/images_original 0
/content/music_genre/Data/images_original/jazz 99
/content/music_genre/Data/images_original/blues 100
/content/music_genre/Data/images_original/hiphop 100
/content/music_genre/Data/images_original/pop 100
/content/music_genre/Data/images_original/classical 100
/content/music_genre/Data/images_original/reggae 100
/content/music_genre/Data/images_original/metal 100
```

```
/content/music_genre/Data/images_original/disco 100
/content/music_genre/Data/images_original/country 100
/content/music_genre/Data/images_original/rock 100
/content/music_genre/Data/genres_original 0
/content/music_genre/Data/genres_original/jazz 100
/content/music_genre/Data/genres_original/blues 100
/content/music_genre/Data/genres_original/hiphop 100
/content/music_genre/Data/genres_original/pop 100
/content/music_genre/Data/genres_original/classical 100
/content/music_genre/Data/genres_original/reggae 100
/content/music_genre/Data/genres_original/metal 100
/content/music_genre/Data/genres_original/disco 100
/content/music_genre/Data/genres_original/country 100
/content/music_genre/Data/genres_original/rock 100
```

[ ]:
```
import glob
import os

# Path to audio files
audio_path = "/content/music_genre/Data/genres_original"

# Get genre folders
genres = [g for g in os.listdir(audio_path) if os.path.isdir(os.path.
 ↪join(audio_path, g))]
print("Genres found:", genres)

filepaths = []
labels = []

for genre in genres:
    files = glob.glob(os.path.join(audio_path, genre, "*.wav"))
    for f in files:
        filepaths.append(f)
        labels.append(genre)

print(f"Found {len(filepaths)} audio files.")
```

```
Genres found: ['jazz', 'blues', 'hiphop', 'pop', 'classical', 'reggae', 'metal',
'disco', 'country', 'rock']
Found 1000 audio files.
```

Audio Feature Extraction (MFCC) – Tabular Approach

[ ]:
```
import librosa
import numpy as np
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.metrics import classification_report, accuracy_score
import warnings
warnings.filterwarnings('ignore')

#  Extract MFCC Features
X = []
y = []

for idx, filepath in enumerate(filepaths):
    try:
        audio, sr = librosa.load(filepath, duration=30)  # load 30 sec clip
        mfccs = librosa.feature.mfcc(y=audio, sr=sr, n_mfcc=20)
        mfccs_mean = np.mean(mfccs.T, axis=0)  # mean over time
        X.append(mfccs_mean)
        y.append(labels[idx])
    except Exception as e:
        print(f"Error processing {filepath}: {e}")

X = np.array(X)
y = np.array(y)

print("Feature matrix shape:", X.shape)
print("Labels count:", len(y))
```

```
Error processing /content/music_genre/Data/genres_original/jazz/jazz.00054.wav:
Feature matrix shape: (999, 20)
Labels count: 999
```

[ ]:
```
#  Encode Labels
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

#  Scale Features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

#  Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y_encoded, test_size=0.2, random_state=42
)
```

Train a Classifier

[ ]:
```
#  Train Classifier (Random Forest)
clf = RandomForestClassifier(n_estimators=200, random_state=42)
clf.fit(X_train, y_train)
```

[ ]:
```
RandomForestClassifier(n_estimators=200, random_state=42)
```

Evaluation

[ ]:
```
y_pred = clf.predict(X_test)
print("\nAccuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred,␣
↪target_names=label_encoder.classes_))
```

```
Accuracy: 0.625

Classification Report:
               precision    recall  f1-score   support

       blues       0.55      0.55      0.55        11
   classical       0.80      0.80      0.80        15
     country       0.46      0.50      0.48        22
       disco       0.50      0.68      0.58        19
      hiphop       0.62      0.54      0.58        24
        jazz       0.76      0.62      0.68        21
       metal       0.62      0.72      0.67        18
         pop       0.76      0.83      0.79        23
      reggae       0.62      0.67      0.64        27
        rock       0.64      0.35      0.45        20

    accuracy                           0.62       200
   macro avg       0.63      0.63      0.62       200
weighted avg       0.63      0.62      0.62       200
```

Generate Spectrogram Images – CNN Approach

[ ]:
```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout
from tensorflow.keras.models import Model

#  Paths
img_dir = "/content/music_genre/Data/images_original"

# Data Generator with Augmentation
datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True
```

```
)

train_gen = datagen.flow_from_directory(
    img_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='training'
)

val_gen = datagen.flow_from_directory(
    img_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='validation'
)
```

```
Found 800 images belonging to 10 classes.
Found 199 images belonging to 10 classes.
```

[ ]:
```
# Load MobileNetV2 Pretrained Model
base_model = MobileNetV2(weights='imagenet', include_top=False,␣
 ↪input_shape=(224, 224, 3))
base_model.trainable = False  # freeze base layers
```

[ ]:
```
#  Build Model
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dropout(0.3)(x)
predictions = Dense(train_gen.num_classes, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=predictions)
```

[ ]:
```
# Compile
model.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])

#  Train
history = model.fit(
    train_gen,
    validation_data=val_gen,
    epochs=5
)
```

```
Epoch 1/5
25/25                 34s 919ms/step -
accuracy: 0.1312 - loss: 2.5960 - val_accuracy: 0.2915 - val_loss: 1.9250
```

```
Epoch 2/5
25/25              14s 567ms/step -
accuracy: 0.2975 - loss: 1.9927 - val_accuracy: 0.4472 - val_loss: 1.6598
Epoch 3/5
25/25              14s 565ms/step -
accuracy: 0.3894 - loss: 1.6612 - val_accuracy: 0.4623 - val_loss: 1.5527
Epoch 4/5
25/25              15s 623ms/step -
accuracy: 0.3993 - loss: 1.6027 - val_accuracy: 0.4925 - val_loss: 1.4535
Epoch 5/5
25/25              14s 565ms/step -
accuracy: 0.5046 - loss: 1.4637 - val_accuracy: 0.4975 - val_loss: 1.4372
```

[ ]:
```python
#  Evaluate
loss, acc = model.evaluate(val_gen)
print(f"\nValidation Accuracy: {acc:.2f}")
```

```
7/7             4s 513ms/step -
accuracy: 0.5191 - loss: 1.3870

Validation Accuracy: 0.50
```

[ ]:
```python

```

Task 7: Sales Forecasting Description

[ ]:
```python
import pandas as pd
import zipfile

# Load dataset
# Extract the zip file
with zipfile.ZipFile('/content/wallmart.zip', 'r') as zip_ref:
    zip_ref.extractall('/content/')

# Load the training data
df = pd.read_csv("/content/train.csv")
```

Data Preparation

[ ]:
```python
# Convert date column to datetime
df['Date'] = pd.to_datetime(df['Date'])

# Sort by date for time series
df = df.sort_values(by='Date')
```

Feature Engineering

```
# Time-based features
df['Month'] = df['Date'].dt.month
df['Week'] = df['Date'].dt.isocalendar().week
df['DayOfWeek'] = df['Date'].dt.dayofweek

# Lag features
df['Sales_Lag_1'] = df.groupby(['Store','Dept'])['Weekly_Sales'].shift(1)
df['Sales_Lag_2'] = df.groupby(['Store','Dept'])['Weekly_Sales'].shift(2)

# Rolling averages
df['Rolling_Mean_4'] = df.groupby(['Store','Dept'])['Weekly_Sales'].shift(1).
↪rolling(4).mean()
```

Handle Missing Values

```
#Lag and rolling averages will create NaN for the first few rows → drop them:
df = df.dropna()
```

Train-Test Split (Time-Aware)

```
#Use time-aware split so the future is not leaked into the past
from sklearn.model_selection import TimeSeriesSplit

# Example split
tscv = TimeSeriesSplit(n_splits=5)
```

Model 1 — Baseline Linear Regression

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import numpy as np

features = ['Month', 'Week', 'DayOfWeek', 'Sales_Lag_1', 'Sales_Lag_2',␣
↪'Rolling_Mean_4']
target = 'Weekly_Sales'

X = df[features]
y = df[target]

# Use last 20% of data as test
split_point = int(len(X)*0.8)
X_train, X_test = X[:split_point], X[split_point:]
y_train, y_test = y[:split_point], y[split_point:]

model_lr = LinearRegression()
model_lr.fit(X_train, y_train)
y_pred_lr = model_lr.predict(X_test)
```

```
rmse_lr = np.sqrt(mean_squared_error(y_test, y_pred_lr))
print("Linear Regression RMSE:", rmse_lr)
```

```
Linear Regression RMSE: 3464.0457885246215
```

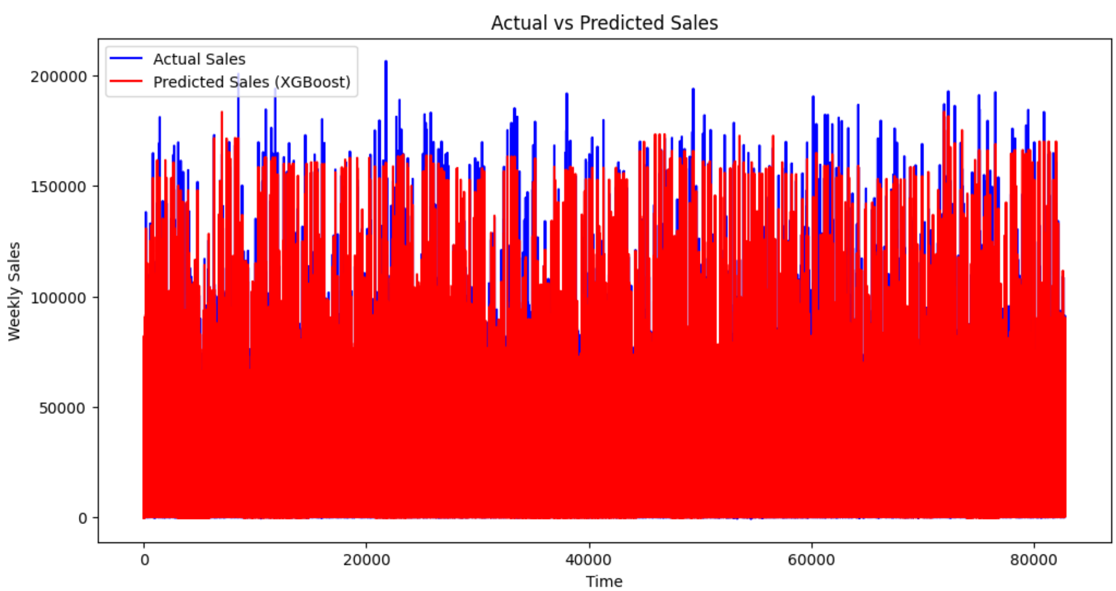
Model 2 — XGBoost

[ ]:
```
from xgboost import XGBRegressor

model_xgb = XGBRegressor(n_estimators=200, learning_rate=0.1, max_depth=6)
model_xgb.fit(X_train, y_train)
y_pred_xgb = model_xgb.predict(X_test)

rmse_xgb = np.sqrt(mean_squared_error(y_test, y_pred_xgb))
print("XGBoost RMSE:", rmse_xgb)
```

```
XGBoost RMSE: 3220.954432468477
```

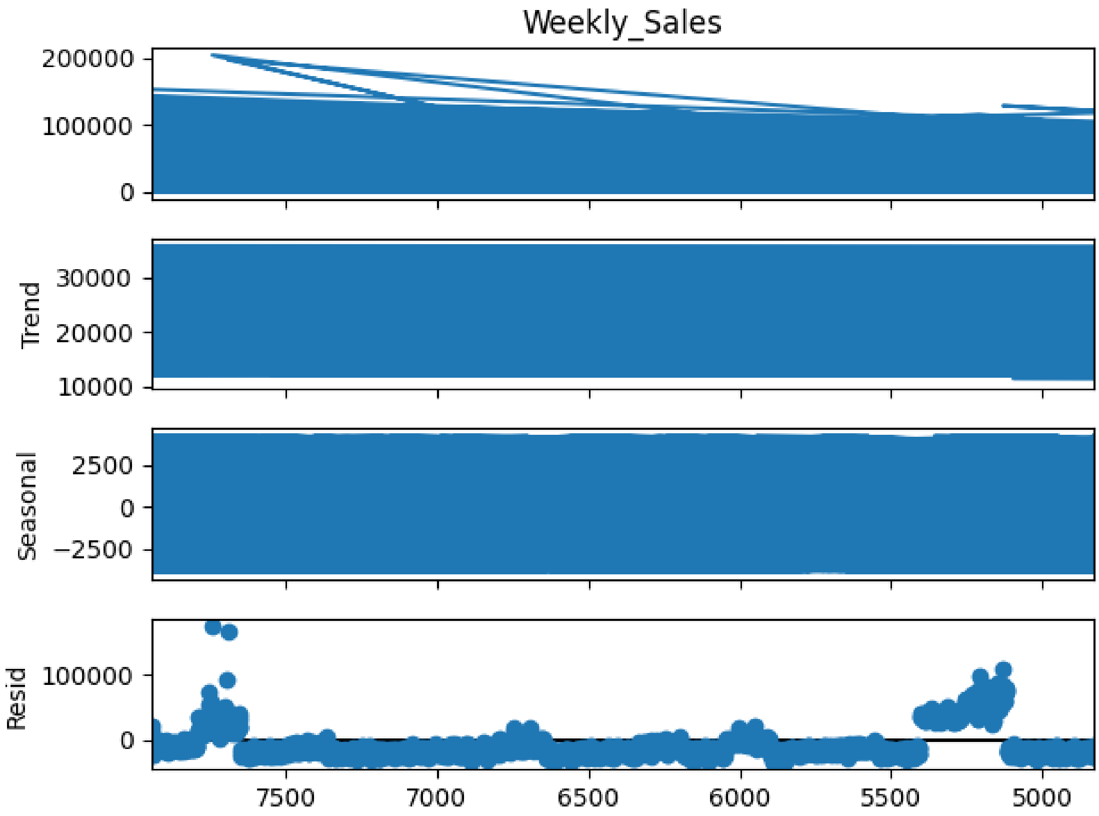
Visualization — Actual vs Predicted

[ ]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(12,6))
plt.plot(y_test.values, label='Actual Sales', color='blue')
plt.plot(y_pred_xgb, label='Predicted Sales (XGBoost)', color='red')
plt.legend()
plt.title('Actual vs Predicted Sales')
plt.xlabel('Time')
plt.ylabel('Weekly Sales')
plt.show()
```



Seasonal Decomposition

[ ]:
```
from statsmodels.tsa.seasonal import seasonal_decompose

df_store = df[df['Store'] == 1]
result = seasonal_decompose(df_store['Weekly_Sales'], model='additive',␣
 ↪period=52)
result.plot()
plt.show()
```



[ ]:
```

```

Task 8: Traffic Sign Recognition Description

[ ]:
```
#Libraries
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

```python
from tensorflow.keras.utils import to_categorical
```

Unzip the Data

[ ]:
```python
import zipfile

# Define the path to the zip file
zip_path = '/content/drive/MyDrive/traffic.zip'
extract_path = '/content/drive/MyDrive/traffic/'

# Unzipping the file
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_path)

print("Dataset unzipped successfully.")
```

```
Dataset unzipped successfully.
```

[ ]:
```python
# List the contents of the extracted folder
os.listdir(extract_path)
```

[ ]:
```
['Meta.csv',
 'Meta',
 'Test.csv',
 'Test',
 'Train.csv',
 'Train',
 'meta',
 'test',
 'train']
```

Load the Data

[ ]:
```python
# Path to dataset
extract_path = '/content/drive/MyDrive/traffic/'

# Paths to CSV files
train_csv = os.path.join(extract_path, 'Train.csv')
test_csv = os.path.join(extract_path, 'Test.csv')

# Read CSVs
train_df = pd.read_csv(train_csv)
test_df = pd.read_csv(test_csv)

# Remove any extra spaces in column names (important!)
train_df.columns = train_df.columns.str.strip()
test_df.columns = test_df.columns.str.strip()
```

```
print(train_df.head())
```

```
   Width  Height  Roi.X1  Roi.Y1  Roi.X2  Roi.Y2  ClassId  \
0     27      26       5       5      22      20       20
1     28      27       5       6      23      22       20
2     29      26       6       5      24      21       20
3     28      27       5       6      23      22       20
4     28      26       5       5      23      21       20

                            Path
0  Train/20/00020_00000_00000.png
1  Train/20/00020_00000_00001.png
2  Train/20/00020_00000_00002.png
3  Train/20/00020_00000_00003.png
4  Train/20/00020_00000_00004.png
```

Load & Preprocess Images

```
[ ]: IMG_SIZE = 32

# Load training images
train_images = []
train_labels = []

for idx, row in train_df.iterrows():
    img_path = os.path.join(extract_path, row['Path'])
    img = cv2.imread(img_path)
    img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
    train_images.append(img)
    train_labels.append(row['ClassId'])

X_train = np.array(train_images) / 255.0  # normalize
y_train = to_categorical(np.array(train_labels))
```

Train-validation split

```
[ ]: X_train, X_val, y_train, y_val = train_test_split(
    X_train, y_train, test_size=0.2, random_state=42
)

print("Train shape:", X_train.shape)
print("Validation shape:", X_val.shape)
```

```
Train shape: (31367, 32, 32, 3)
Validation shape: (7842, 32, 32, 3)
```

Build & compile CNN

[ ]:
```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,␣
↪Dropout

model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(IMG_SIZE, IMG_SIZE, 3)),
    MaxPooling2D(2,2),
    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(y_train.shape[1], activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy',␣
↪metrics=['accuracy'])
model.summary()
```

```
/usr/local/lib/python3.11/dist-
packages/keras/src/layers/convolutional/base_conv.py:113: UserWarning: Do not
pass an `input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in the model
instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_1 (Conv2D)	(None, 13, 13, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 64)	0
flatten (Flatten)	(None, 2304)	0
dense (Dense)	(None, 128)	295,040
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 43)	5,547

Total params: 319,979 (1.22 MB)

Trainable params: 319,979 (1.22 MB)

Non-trainable params: 0 (0.00 B)

Train the Model

```
[ ]: history = model.fit(
         X_train, y_train,
         validation_data=(X_val, y_val),
         epochs=3,
         batch_size=32
     )
```

```
Epoch 1/3
981/981              17s 9ms/step -
accuracy: 0.3413 - loss: 2.4550 - val_accuracy: 0.9118 - val_loss: 0.4010
Epoch 2/3
981/981              5s 5ms/step -
accuracy: 0.8233 - loss: 0.5659 - val_accuracy: 0.9651 - val_loss: 0.1538
Epoch 3/3
981/981              4s 4ms/step -
accuracy: 0.8950 - loss: 0.3371 - val_accuracy: 0.9776 - val_loss: 0.0964
```

Test Data Evaluation

```
[ ]: test_images = []
     test_labels = []

     for idx, row in test_df.iterrows():
         img_path = os.path.join(extract_path, row['Path'])
         img = cv2.imread(img_path)
         img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
         test_images.append(img)
         test_labels.append(row['ClassId'])

     X_test = np.array(test_images) / 255.0
     y_test = to_categorical(np.array(test_labels))

     loss, acc = model.evaluate(X_test, y_test)
     print(f"Test Accuracy: {acc*100:.2f}%")
```

```
395/395              2s 6ms/step -
accuracy: 0.9318 - loss: 0.2764
Test Accuracy: 93.10%
```

Add Data Augmentation

[ ]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(
    rotation_range=10,
    zoom_range=0.1,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    horizontal_flip=False,
    fill_mode='nearest'
)

datagen.fit(X_train)

# Retrain the same CNN with augmentation
history_aug = model.fit(
    datagen.flow(X_train, y_train, batch_size=32),
    validation_data=(X_val, y_val),
    epochs=10
)
```

```
Epoch 1/10
```

```
/usr/local/lib/python3.11/dist-
packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121:
UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in
its constructor. `**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will be
ignored.
  self._warn_if_super_not_called()
```

```
981/981              26s 24ms/step -
accuracy: 0.6251 - loss: 1.3288 - val_accuracy: 0.9652 - val_loss: 0.1443
Epoch 2/10
981/981              40s 25ms/step -
accuracy: 0.7540 - loss: 0.7829 - val_accuracy: 0.9759 - val_loss: 0.1234
Epoch 3/10
981/981              29s 29ms/step -
accuracy: 0.7985 - loss: 0.6382 - val_accuracy: 0.9753 - val_loss: 0.1136
Epoch 4/10
981/981              23s 23ms/step -
accuracy: 0.8245 - loss: 0.5410 - val_accuracy: 0.9790 - val_loss: 0.0944
Epoch 5/10
981/981              38s 20ms/step -
accuracy: 0.8524 - loss: 0.4723 - val_accuracy: 0.9856 - val_loss: 0.0658
Epoch 6/10
981/981              24s 24ms/step -
```

```
accuracy: 0.8602 - loss: 0.4257 - val_accuracy: 0.9839 - val_loss: 0.0689
Epoch 7/10
981/981              22s 22ms/step -
accuracy: 0.8788 - loss: 0.3758 - val_accuracy: 0.9890 - val_loss: 0.0479
Epoch 8/10
981/981              20s 20ms/step -
accuracy: 0.8897 - loss: 0.3459 - val_accuracy: 0.9888 - val_loss: 0.0482
Epoch 9/10
981/981              21s 22ms/step -
accuracy: 0.9003 - loss: 0.3235 - val_accuracy: 0.9893 - val_loss: 0.0431
Epoch 10/10
981/981              21s 21ms/step -
accuracy: 0.9026 - loss: 0.3022 - val_accuracy: 0.9911 - val_loss: 0.0345
```

MobileNetV2 Transfer Learning

[ ]:
```python
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Model
from tensorflow.keras.layers import GlobalAveragePooling2D

# Load MobileNetV2 without top layers
base_model = MobileNetV2(weights='imagenet', include_top=False,
 ↪input_shape=(IMG_SIZE, IMG_SIZE, 3))
base_model.trainable = False  # freeze base model

# Add custom layers on top
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(y_train.shape[1], activation='softmax')(x)

mobilenet_model = Model(inputs=base_model.input, outputs=predictions)

mobilenet_model.compile(optimizer='adam', loss='categorical_crossentropy',
 ↪metrics=['accuracy'])

mobilenet_model.summary()
```

```
/tmp/ipython-input-859176245.py:6: UserWarning: `input_shape` is undefined or
non-square, or `rows` is not in [96, 128, 160, 192, 224]. Weights for input
shape (224, 224) will be loaded as the default.
  base_model = MobileNetV2(weights='imagenet', include_top=False,
input_shape=(IMG_SIZE, IMG_SIZE, 3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applicatio
ns/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_224_no_top.h
5
9406464/9406464              0s
```

0us/step

Model: "functional_1"

Layer (type)	Output Shape	Param #	Connected to
input_layer_1 (InputLayer)	(None, 32, 32, 3)	0	-
Conv1 (Conv2D)	(None, 16, 16, 32)	864	input_layer_1[0]…
bn_Conv1 (BatchNormalizatio…	(None, 16, 16, 32)	128	Conv1[0][0]
Conv1_relu (ReLU)	(None, 16, 16, 32)	0	bn_Conv1[0][0]
expanded_conv_dept… (DepthwiseConv2D)	(None, 16, 16, 32)	288	Conv1_relu[0][0]
expanded_conv_dept… (BatchNormalizatio…	(None, 16, 16, 32)	128	expanded_conv_de…
expanded_conv_dept… (ReLU)	(None, 16, 16, 32)	0	expanded_conv_de…
expanded_conv_proj… (Conv2D)	(None, 16, 16, 16)	512	expanded_conv_de…
expanded_conv_proj… (BatchNormalizatio…	(None, 16, 16, 16)	64	expanded_conv_pr…
block_1_expand (Conv2D)	(None, 16, 16, 96)	1,536	expanded_conv_pr…
block_1_expand_BN (BatchNormalizatio…	(None, 16, 16, 96)	384	block_1_expand[0…
block_1_expand_relu (ReLU)	(None, 16, 16, 96)	0	block_1_expand_B…
block_1_pad (ZeroPadding2D)	(None, 17, 17, 96)	0	block_1_expand_r…
block_1_depthwise (DepthwiseConv2D)	(None, 8, 8, 96)	864	block_1_pad[0][0]

block_1_depthwise_… (BatchNormalizatio…	(None, 8, 8, 96)	384	block_1_depthwis…
block_1_depthwise_… (ReLU)	(None, 8, 8, 96)	0	block_1_depthwis…
block_1_project (Conv2D)	(None, 8, 8, 24)	2,304	block_1_depthwis…
block_1_project_BN (BatchNormalizatio…	(None, 8, 8, 24)	96	block_1_project[…
block_2_expand (Conv2D)	(None, 8, 8, 144)	3,456	block_1_project_…
block_2_expand_BN (BatchNormalizatio…	(None, 8, 8, 144)	576	block_2_expand[0…
block_2_expand_relu (ReLU)	(None, 8, 8, 144)	0	block_2_expand_B…
block_2_depthwise (DepthwiseConv2D)	(None, 8, 8, 144)	1,296	block_2_expand_r…
block_2_depthwise_… (BatchNormalizatio…	(None, 8, 8, 144)	576	block_2_depthwis…
block_2_depthwise_… (ReLU)	(None, 8, 8, 144)	0	block_2_depthwis…
block_2_project (Conv2D)	(None, 8, 8, 24)	3,456	block_2_depthwis…
block_2_project_BN (BatchNormalizatio…	(None, 8, 8, 24)	96	block_2_project[…
block_2_add (Add)	(None, 8, 8, 24)	0	block_1_project_… block_2_project_…
block_3_expand (Conv2D)	(None, 8, 8, 144)	3,456	block_2_add[0][0]
block_3_expand_BN (BatchNormalizatio…	(None, 8, 8, 144)	576	block_3_expand[0…
block_3_expand_relu (ReLU)	(None, 8, 8, 144)	0	block_3_expand_B…

block_3_pad (ZeroPadding2D)	(None, 9, 9, 144)	0	block_3_expand_r…
block_3_depthwise (DepthwiseConv2D)	(None, 4, 4, 144)	1,296	block_3_pad[0][0]
block_3_depthwise_… (BatchNormalizatio…	(None, 4, 4, 144)	576	block_3_depthwis…
block_3_depthwise_… (ReLU)	(None, 4, 4, 144)	0	block_3_depthwis…
block_3_project (Conv2D)	(None, 4, 4, 32)	4,608	block_3_depthwis…
block_3_project_BN (BatchNormalizatio…	(None, 4, 4, 32)	128	block_3_project[…
block_4_expand (Conv2D)	(None, 4, 4, 192)	6,144	block_3_project_…
block_4_expand_BN (BatchNormalizatio…	(None, 4, 4, 192)	768	block_4_expand[0…
block_4_expand_relu (ReLU)	(None, 4, 4, 192)	0	block_4_expand_B…
block_4_depthwise (DepthwiseConv2D)	(None, 4, 4, 192)	1,728	block_4_expand_r…
block_4_depthwise_… (BatchNormalizatio…	(None, 4, 4, 192)	768	block_4_depthwis…
block_4_depthwise_… (ReLU)	(None, 4, 4, 192)	0	block_4_depthwis…
block_4_project (Conv2D)	(None, 4, 4, 32)	6,144	block_4_depthwis…
block_4_project_BN (BatchNormalizatio…	(None, 4, 4, 32)	128	block_4_project[…
block_4_add (Add)	(None, 4, 4, 32)	0	block_3_project_… block_4_project_…
block_5_expand (Conv2D)	(None, 4, 4, 192)	6,144	block_4_add[0][0]

block_5_expand_BN (BatchNormalizatio…	(None, 4, 4, 192)	768	block_5_expand[0…
block_5_expand_relu (ReLU)	(None, 4, 4, 192)	0	block_5_expand_B…
block_5_depthwise (DepthwiseConv2D)	(None, 4, 4, 192)	1,728	block_5_expand_r…
block_5_depthwise_… (BatchNormalizatio…	(None, 4, 4, 192)	768	block_5_depthwis…
block_5_depthwise_… (ReLU)	(None, 4, 4, 192)	0	block_5_depthwis…
block_5_project (Conv2D)	(None, 4, 4, 32)	6,144	block_5_depthwis…
block_5_project_BN (BatchNormalizatio…	(None, 4, 4, 32)	128	block_5_project[…
block_5_add (Add)	(None, 4, 4, 32)	0	block_4_add[0][0… block_5_project_…
block_6_expand (Conv2D)	(None, 4, 4, 192)	6,144	block_5_add[0][0]
block_6_expand_BN (BatchNormalizatio…	(None, 4, 4, 192)	768	block_6_expand[0…
block_6_expand_relu (ReLU)	(None, 4, 4, 192)	0	block_6_expand_B…
block_6_pad (ZeroPadding2D)	(None, 5, 5, 192)	0	block_6_expand_r…
block_6_depthwise (DepthwiseConv2D)	(None, 2, 2, 192)	1,728	block_6_pad[0][0]
block_6_depthwise_… (BatchNormalizatio…	(None, 2, 2, 192)	768	block_6_depthwis…
block_6_depthwise_… (ReLU)	(None, 2, 2, 192)	0	block_6_depthwis…
block_6_project (Conv2D)	(None, 2, 2, 64)	12,288	block_6_depthwis…

block_6_project_BN (BatchNormalizatio…	(None, 2, 2, 64)	256	block_6_project[…
block_7_expand (Conv2D)	(None, 2, 2, 384)	24,576	block_6_project_…
block_7_expand_BN (BatchNormalizatio…	(None, 2, 2, 384)	1,536	block_7_expand[0…
block_7_expand_relu (ReLU)	(None, 2, 2, 384)	0	block_7_expand_B…
block_7_depthwise (DepthwiseConv2D)	(None, 2, 2, 384)	3,456	block_7_expand_r…
block_7_depthwise_… (BatchNormalizatio…	(None, 2, 2, 384)	1,536	block_7_depthwis…
block_7_depthwise_… (ReLU)	(None, 2, 2, 384)	0	block_7_depthwis…
block_7_project (Conv2D)	(None, 2, 2, 64)	24,576	block_7_depthwis…
block_7_project_BN (BatchNormalizatio…	(None, 2, 2, 64)	256	block_7_project[…
block_7_add (Add)	(None, 2, 2, 64)	0	block_6_project_… block_7_project_…
block_8_expand (Conv2D)	(None, 2, 2, 384)	24,576	block_7_add[0][0]
block_8_expand_BN (BatchNormalizatio…	(None, 2, 2, 384)	1,536	block_8_expand[0…
block_8_expand_relu (ReLU)	(None, 2, 2, 384)	0	block_8_expand_B…
block_8_depthwise (DepthwiseConv2D)	(None, 2, 2, 384)	3,456	block_8_expand_r…
block_8_depthwise_… (BatchNormalizatio…	(None, 2, 2, 384)	1,536	block_8_depthwis…
block_8_depthwise_… (ReLU)	(None, 2, 2, 384)	0	block_8_depthwis…

block_8_project (Conv2D)	(None, 2, 2, 64)	24,576	block_8_depthwis…
block_8_project_BN (BatchNormalizatio…	(None, 2, 2, 64)	256	block_8_project[…
block_8_add (Add)	(None, 2, 2, 64)	0	block_7_add[0][0… block_8_project_…
block_9_expand (Conv2D)	(None, 2, 2, 384)	24,576	block_8_add[0][0]
block_9_expand_BN (BatchNormalizatio…	(None, 2, 2, 384)	1,536	block_9_expand[0…
block_9_expand_relu (ReLU)	(None, 2, 2, 384)	0	block_9_expand_B…
block_9_depthwise (DepthwiseConv2D)	(None, 2, 2, 384)	3,456	block_9_expand_r…
block_9_depthwise_… (BatchNormalizatio…	(None, 2, 2, 384)	1,536	block_9_depthwis…
block_9_depthwise_… (ReLU)	(None, 2, 2, 384)	0	block_9_depthwis…
block_9_project (Conv2D)	(None, 2, 2, 64)	24,576	block_9_depthwis…
block_9_project_BN (BatchNormalizatio…	(None, 2, 2, 64)	256	block_9_project[…
block_9_add (Add)	(None, 2, 2, 64)	0	block_8_add[0][0… block_9_project_…
block_10_expand (Conv2D)	(None, 2, 2, 384)	24,576	block_9_add[0][0]
block_10_expand_BN (BatchNormalizatio…	(None, 2, 2, 384)	1,536	block_10_expand[…
block_10_expand_re… (ReLU)	(None, 2, 2, 384)	0	block_10_expand_…
block_10_depthwise (DepthwiseConv2D)	(None, 2, 2, 384)	3,456	block_10_expand_…

block_10_depthwise… (BatchNormalizatio…	(None, 2, 2, 384)	1,536	block_10_depthwi…
block_10_depthwise… (ReLU)	(None, 2, 2, 384)	0	block_10_depthwi…
block_10_project (Conv2D)	(None, 2, 2, 96)	36,864	block_10_depthwi…
block_10_project_BN (BatchNormalizatio…	(None, 2, 2, 96)	384	block_10_project…
block_11_expand (Conv2D)	(None, 2, 2, 576)	55,296	block_10_project…
block_11_expand_BN (BatchNormalizatio…	(None, 2, 2, 576)	2,304	block_11_expand[…
block_11_expand_re… (ReLU)	(None, 2, 2, 576)	0	block_11_expand_…
block_11_depthwise (DepthwiseConv2D)	(None, 2, 2, 576)	5,184	block_11_expand_…
block_11_depthwise… (BatchNormalizatio…	(None, 2, 2, 576)	2,304	block_11_depthwi…
block_11_depthwise… (ReLU)	(None, 2, 2, 576)	0	block_11_depthwi…
block_11_project (Conv2D)	(None, 2, 2, 96)	55,296	block_11_depthwi…
block_11_project_BN (BatchNormalizatio…	(None, 2, 2, 96)	384	block_11_project…
block_11_add (Add)	(None, 2, 2, 96)	0	block_10_project… block_11_project…
block_12_expand (Conv2D)	(None, 2, 2, 576)	55,296	block_11_add[0][…
block_12_expand_BN (BatchNormalizatio…	(None, 2, 2, 576)	2,304	block_12_expand[…
block_12_expand_re… (ReLU)	(None, 2, 2, 576)	0	block_12_expand_…

block_12_depthwise (DepthwiseConv2D)	(None, 2, 2, 576)	5,184	block_12_expand_…
block_12_depthwise… (BatchNormalizatio…	(None, 2, 2, 576)	2,304	block_12_depthwi…
block_12_depthwise… (ReLU)	(None, 2, 2, 576)	0	block_12_depthwi…
block_12_project (Conv2D)	(None, 2, 2, 96)	55,296	block_12_depthwi…
block_12_project_BN (BatchNormalizatio…	(None, 2, 2, 96)	384	block_12_project…
block_12_add (Add)	(None, 2, 2, 96)	0	block_11_add[0][… block_12_project…
block_13_expand (Conv2D)	(None, 2, 2, 576)	55,296	block_12_add[0][…
block_13_expand_BN (BatchNormalizatio…	(None, 2, 2, 576)	2,304	block_13_expand[…
block_13_expand_re… (ReLU)	(None, 2, 2, 576)	0	block_13_expand_…
block_13_pad (ZeroPadding2D)	(None, 3, 3, 576)	0	block_13_expand_…
block_13_depthwise (DepthwiseConv2D)	(None, 1, 1, 576)	5,184	block_13_pad[0][…
block_13_depthwise… (BatchNormalizatio…	(None, 1, 1, 576)	2,304	block_13_depthwi…
block_13_depthwise… (ReLU)	(None, 1, 1, 576)	0	block_13_depthwi…
block_13_project (Conv2D)	(None, 1, 1, 160)	92,160	block_13_depthwi…
block_13_project_BN (BatchNormalizatio…	(None, 1, 1, 160)	640	block_13_project…
block_14_expand (Conv2D)	(None, 1, 1, 960)	153,600	block_13_project…

block_14_expand_BN (BatchNormalizatio…	(None, 1, 1, 960)	3,840	block_14_expand[…
block_14_expand_re… (ReLU)	(None, 1, 1, 960)	0	block_14_expand_…
block_14_depthwise (DepthwiseConv2D)	(None, 1, 1, 960)	8,640	block_14_expand_…
block_14_depthwise… (BatchNormalizatio…	(None, 1, 1, 960)	3,840	block_14_depthwi…
block_14_depthwise… (ReLU)	(None, 1, 1, 960)	0	block_14_depthwi…
block_14_project (Conv2D)	(None, 1, 1, 160)	153,600	block_14_depthwi…
block_14_project_BN (BatchNormalizatio…	(None, 1, 1, 160)	640	block_14_project…
block_14_add (Add)	(None, 1, 1, 160)	0	block_13_project… block_14_project…
block_15_expand (Conv2D)	(None, 1, 1, 960)	153,600	block_14_add[0][…
block_15_expand_BN (BatchNormalizatio…	(None, 1, 1, 960)	3,840	block_15_expand[…
block_15_expand_re… (ReLU)	(None, 1, 1, 960)	0	block_15_expand_…
block_15_depthwise (DepthwiseConv2D)	(None, 1, 1, 960)	8,640	block_15_expand_…
block_15_depthwise… (BatchNormalizatio…	(None, 1, 1, 960)	3,840	block_15_depthwi…
block_15_depthwise… (ReLU)	(None, 1, 1, 960)	0	block_15_depthwi…
block_15_project (Conv2D)	(None, 1, 1, 160)	153,600	block_15_depthwi…
block_15_project_BN (BatchNormalizatio…	(None, 1, 1, 160)	640	block_15_project…

block_15_add (Add)	(None, 1, 1, 160)	0	block_14_add[0][… block_15_project…
block_16_expand (Conv2D)	(None, 1, 1, 960)	153,600	block_15_add[0][…
block_16_expand_BN (BatchNormalizatio…	(None, 1, 1, 960)	3,840	block_16_expand[…
block_16_expand_re… (ReLU)	(None, 1, 1, 960)	0	block_16_expand_…
block_16_depthwise (DepthwiseConv2D)	(None, 1, 1, 960)	8,640	block_16_expand_…
block_16_depthwise… (BatchNormalizatio…	(None, 1, 1, 960)	3,840	block_16_depthwi…
block_16_depthwise… (ReLU)	(None, 1, 1, 960)	0	block_16_depthwi…
block_16_project (Conv2D)	(None, 1, 1, 320)	307,200	block_16_depthwi…
block_16_project_BN (BatchNormalizatio…	(None, 1, 1, 320)	1,280	block_16_project…
Conv_1 (Conv2D)	(None, 1, 1, 1280)	409,600	block_16_project…
Conv_1_bn (BatchNormalizatio…	(None, 1, 1, 1280)	5,120	Conv_1[0][0]
out_relu (ReLU)	(None, 1, 1, 1280)	0	Conv_1_bn[0][0]
global_average_poo… (GlobalAveragePool…	(None, 1280)	0	out_relu[0][0]
dense_2 (Dense)	(None, 128)	163,968	global_average_p…
dropout_1 (Dropout)	(None, 128)	0	dense_2[0][0]
dense_3 (Dense)	(None, 43)	5,547	dropout_1[0][0]

Total params: 2,427,499 (9.26 MB)

Trainable params: 169,515 (662.17 KB)

Non-trainable params: 2,257,984 (8.61 MB)

Train MobileNetV2

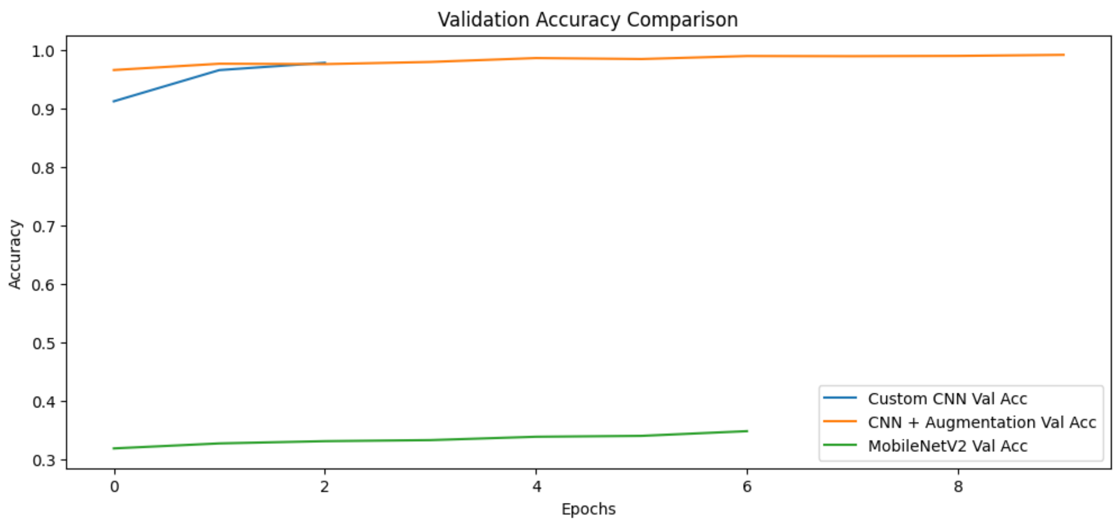
[ ]:
```
history_mobilenet = mobilenet_model.fit(
    datagen.flow(X_train, y_train, batch_size=32),
    validation_data=(X_val, y_val),
    epochs=7
)
```

```
Epoch 1/7
981/981              27s 28ms/step -
accuracy: 0.2825 - loss: 2.4340 - val_accuracy: 0.3192 - val_loss: 2.2776
Epoch 2/7
981/981              28s 28ms/step -
accuracy: 0.2860 - loss: 2.3973 - val_accuracy: 0.3277 - val_loss: 2.2460
Epoch 3/7
981/981              27s 28ms/step -
accuracy: 0.2932 - loss: 2.3779 - val_accuracy: 0.3315 - val_loss: 2.2198
Epoch 4/7
981/981              26s 26ms/step -
accuracy: 0.2984 - loss: 2.3605 - val_accuracy: 0.3333 - val_loss: 2.2042
Epoch 5/7
981/981              41s 26ms/step -
accuracy: 0.2972 - loss: 2.3432 - val_accuracy: 0.3391 - val_loss: 2.1873
Epoch 6/7
981/981              24s 25ms/step -
accuracy: 0.3020 - loss: 2.3190 - val_accuracy: 0.3406 - val_loss: 2.1776
Epoch 7/7
981/981              24s 24ms/step -
accuracy: 0.3061 - loss: 2.3096 - val_accuracy: 0.3486 - val_loss: 2.1603
```

Compare Models

[ ]:
```
import matplotlib.pyplot as plt

def plot_history(histories, titles):
    plt.figure(figsize=(12,5))
    for history, label in zip(histories, titles):
        plt.plot(history.history['val_accuracy'], label=f'{label} Val Acc')
    plt.title('Validation Accuracy Comparison')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
```

```
    plt.legend()
    plt.show()

plot_history(
    [history, history_aug, history_mobilenet],
    ['Custom CNN', 'CNN + Augmentation', 'MobileNetV2']
)
```




```
[ ]: %%shell
cd Elevvo-Pathways-ML-intern/
git add .
git commit -m "Update notebook with tasks"
git push origin HEAD
```