NAME-
SARVAGYA SANJAY(PES1UG22EC913)

CODE

```
[ ] import pandas as pd
```

```
df = pd.read_csv("TMDB  IMDB Movies Dataset.csv")
df = df.head(10000)
df
```

```
df.shape
```

```
(10000, 29)
```

```
[ ] df.columns
```

```
Index(['id', 'title', 'vote_average', 'vote_count', 'status', 'release_date',
       'revenue', 'runtime', 'adult', 'backdrop_path', 'budget', 'homepage',
       'tconst', 'original_language', 'original_title', 'overview',
       'popularity', 'poster_path', 'tagline', 'genres',
       'production_companies', 'production_countries', 'spoken_languages',
       'keywords', 'directors', 'writers', 'averageRating', 'numVotes',
       'cast'],
      dtype='object')
```

```
[ ] columns_to_drop = [
        "id", "tconst", "status", "adult", "backdrop_path",
        "poster_path", "homepage", "overview", "tagline", "production_companies",
        "production_countries", "spoken_languages", "keywords", "writers",
        "averageRating", "numVotes", "vote_average", "vote_count", "popularity"
    ]

    df = df.drop(columns=columns_to_drop)
    # df.shape  # Check the new shape
    df.head()  # Verify remaining columns
```

Preparing Dataset for Training

```
df["hit_or_flop"] = (df["revenue"] >= 2 * df["budget"]).astype(int)
df = df.drop(columns=["budget"])  # We don't need them anymore
df.head()  # Check if the target column is added
```

NAME-
SARVAGYA SANJAY(PES1UG22EC913)

```
[ ] df.to_csv("preprocessed_movies.csv", index=False)
```

```
[ ] df = pd.read_csv("preprocessed_movies.csv")
```

```
import ast

# Convert genres into lists properly
def clean_genres(genre_str):
    if isinstance(genre_str, str):
        try:
            return ast.literal_eval(genre_str)  # Try to convert to list
        except:
            return genre_str.split(", ")  # If it fails, split manually
    return []

df["genres"] = df["genres"].apply(clean_genres)

# Check if genres are now lists
df["genres"].head()
```

	genres
0	[Action, Science Fiction, Adventure]
1	[Adventure, Drama, Science Fiction]
2	[Drama, Action, Crime, Thriller]
3	[Action, Adventure, Fantasy, Science Fiction]
4	[Science Fiction, Action, Adventure]

dtype: object

Splitting the DataSet (80 - 20)

```
[ ] from sklearn.model_selection import train_test_split

    # Define features (X) and target variable (y)
    X = df.drop(columns=["hit_or_flop"])  # Drop the target column
    y = df["hit_or_flop"]  # Target variable (Hit = 1, Flop = 0)

    # Split dataset into train (80%) and test (20%)
    X_train, X_test, y_train, y_test =train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

    # Check shapes
    print("Training set size:", X_train.shape)
    print("Testing set size:", X_test.shape)
```

NAME-
SARVAGYA SANJAY(PES1UG22EC913)

MODEL 1- GRADIENT BOOST

```
[ ] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import LabelEncoder, StandardScaler
    from sklearn.svm import SVC
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
    from sklearn.ensemble import GradientBoostingClassifier
```

```
[ ] from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import StandardScaler, OneHotEncoder

    # Load Dataset
    df = pd.read_csv("preprocessed_movies.csv")

    # Define Features and Target
    features = ["genres", "directors", "cast","revenue"]
    target = "hit_or_flop"

    # Convert Categorical Features into Numerical using One-Hot Encoding
    df_encoded = pd.get_dummies(df[features])

    # Splitting Data
    X_train, X_test, y_train, y_test = train_test_split(df_encoded, df[target], test_size=0.2, random_state=42)

    # Standardizing Features (Only if data is numeric now)
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    print("Preprocessing Completed Successfully!")
```

NAME-
SARVAGYA SANJAY(PES1UG22EC913)

MODEL 1 GRADIENT BOOSTING

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score, classification_report

# Separate features and target
X = df.drop("hit_or_flop", axis=1)
y = df["hit_or_flop"]

# One-Hot Encode non-numeric (categorical) columns
X_encoded = pd.get_dummies(X, drop_first=True)

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

# Initialize and train Gradient Boosting model
gb_model = GradientBoostingClassifier(
    n_estimators=200,
    learning_rate=0.1,
    max_depth=4,
    random_state=42
)
gb_model.fit(X_train, y_train)

# Predict on test data
y_pred_gb = gb_model.predict(X_test)

# Evaluate model
print(f"📊 Gradient Boosting Accuracy: {accuracy_score(y_test, y_pred_gb):.4f}")

print("\n📄 Classification Report:")
print(classification_report(y_test, y_pred_gb))
```

MODEL 2- STATE VECTOR MACHINE

using rbf

```python
from sklearn.svm import SVC

# Train SVM model
svm_model = SVC(kernel='rbf', random_state=42)
svm_model.fit(X_train, y_train)

# Predictions
y_pred_svm = svm_model.predict(X_test)

# Accuracy
svm_accuracy = accuracy_score(y_test, y_pred_svm)
print(f" SVM Accuracy: {svm_accuracy:.4f}")

# Classification Report
print("\n SVM - Classification Report:")
print(classification_report(y_test, y_pred_svm))
```

NAME-
SARVAGYA SANJAY(PES1UG22EC913)

MODEL 3- RANDOM FOREST

```
[ ]
```

```
from sklearn.ensemble import RandomForestClassifier

# Train Random Forest model
rf_model = RandomForestClassifier(
    n_estimators=500,
    max_depth=10,
    random_state=42,
    class_weight="balanced"
)
rf_model.fit(X_train, y_train)

# Predictions
y_pred_rf = rf_model.predict(X_test)

# Accuracy
rf_accuracy = accuracy_score(y_test, y_pred_rf)
print(f"📌 Random Forest Accuracy: {rf_accuracy:.4f}")

# Classification Report
print("\n📊 Random Forest - Classification Report:")
print(classification_report(y_test, y_pred_rf))
```

```
# Accuracy Scores
models = ["Gradient Boosting", "SVM", "Random Forest"]
accuracies = [accuracy_log, svm_accuracy, rf_accuracy]

# Plot Accuracy
plt.figure(figsize=(8, 5))
plt.bar(models, accuracies, color=["blue", "green", "orange"])
plt.xlabel("Model")
plt.ylabel("Accuracy")
plt.title("Model Accuracy Comparison")
plt.ylim(0, 1)  # Accuracy range
plt.show()
```