A Project Report On

Student Stream Prediction

using Machine Learning

Submitted in partial fulfillment of the requirement for the award of the degree

BACHELOR OF SCIENCE (DATA SCIENCE)
from
Marwadi University

Academic Year 2025 – 26

Diya Tilwani (92300566002)
Rahi Ghodke (92300566012)
Rishita Shah (92300566021)

Internal Guide
Jignesh Hirapara

Rajkot-Morbi Road, At &PO :Gauridad, Rajkot 360 003. Gujarat. India.

Faculty of Computer Applications (FoCA)

Certificate

This is to certify that the project work entitled

Student Stream Prediction using Machine Learning

submitted in partial fulfillment of the requirement for the award of the degree of

Bachelor of Science (Data Science)

of the

Marwadi University

is a result of the bonafide work carried out by

Diya Tilwani (92300566002)
Rahi Ghodke(92300566012)
Rishita Shah (92300566021)

during the academic year 2025 – 2026

Faculty Guide	HOD	Dean

DECLARATION

We hereby declare that this project work entitled Student Stream Prediction using Machine Learning and Streamlit is a record done by us.

We also declare that the matter embodied in this project is genuine work done by us and has not been submitted whether to this University or to any other University / Institute for the fulfillment of the requirement of any course of study.

Place:

Date:

Diya Tilwani (92300566002) Signature:_____
Rahi Ghodke (92300566012) Signature:_____
Rishita Shah (92300566021) Signature:_____

ACKNOWLEDGEMENT

It is indeed a great pleasure to express our thanks and gratitude to all those who helped us. No serious and lasting achievement or success one can ever achieve without the help of friendly guidance and co-operation of so many people involved in the work.

We are very thankful to our guide **Jignesh Hirapara,** the person who makes us to follow the right steps during our project work. We express our deep sense of gratitude to for his /her guidance, suggestions and expertise at every stage. A part from that his/her valuable and expertise suggestion during documentation of our report indeed help us a lot.

Thanks to our friend and colleague who have been a source of inspiration and motivation that helped to us during our project work.

We are heartily thankful to the **Dean** of our department **Dr. R. Sridharan** sir and **HoD Dr. Sunil Bajeja** sir for giving us an opportunity to work over this project and for their end-less and great support. And to all other people who directly or indirectly supported and help us to fulfil our task.

Diya Tilwani (92300566002) Signature: _____
Rahi Ghodke (92300566012) Signature: _____
Rishita Shah (92300566021) Signature:_____

CONTENTS

Table Index

Figure Index

Chapter 1: Introduction

Objective of the New System

The objective of this project is to build an intelligent recommendation system that predicts the most suitable academic stream (Science, Commerce, or Arts) for a student using their Grade 10, Grade 12, JEE, and CUET marks.

Problem Definition

Students often face challenges while selecting the right stream for higher education. A wrong choice may affect academic performance and career growth. Currently, such decisions are mostly based on parental or peer opinions. This project provides a data-driven solution by applying machine learning techniques to assist students in choosing the best stream.

Core Components

1. Dataset Preprocessing – Cleaning and handling missing values.
2. Feature Engineering – Converting JEE and CUET scores into comparable scales, creating average scores, and combined indices.
3. Model Training – Using Logistic Regression, Support Vector Machine, and Random Forest Classifier.
4. Evaluation – Measuring accuracy, classification reports, and confusion matrices.
5. Deployment – A user-friendly web application built with Streamlit.

Project Profile

Language: Python
Libraries: pandas, numpy, matplotlib, seaborn, scikit-learn, joblib, streamlit
Dataset: Student academic performance dataset with Grades, JEE, CUET, and Stream labels
Models Tested: Logistic Regression, SVM, Random Forest

Assumptions and Constraints

Assumptions: Dataset contains valid student marks; CUET score optional.
Constraints: Dataset size is limited, may not represent diverse populations.

Advantages and Limitations of the Proposed System

Advantages: Automated, data-driven predictions, user-friendly interface.
Limitations: Accuracy depends on dataset quality, non-academic factors not considered.

Chapter 2: Requirement Determination & Analysis

Requirement Determination

The project requires a dataset containing student grades, JEE, CUET scores, and stream labels, Python environment with ML libraries, and Streamlit for deployment.

Targeted Users

Students, academic counselors, and institutions aiming to guide career choices.

Tool Details

Python for backend; Streamlit for creating interactive prediction interface.

Library Description

pandas, numpy (data handling), matplotlib, seaborn (visualization), scikit-learn (model building), joblib (saving models), streamlit (deployment).

Chapter 3: System Design

Flowchart / Algorithm with steps

1. Load dataset
2. Data cleaning
3. Feature engineering
4. Train/test split
5. Train models
6. Evaluate models
7. Select best model
8. Deploy using Streamlit

Dataset Design

Attributes:
Name, Age, Gender, City, Grade 10, Grade 12, JEE, CUET, Stream.

Derived attributes:
JEE_100, CUET_100, Academic Average, Entrance Average, Combined Score.

Details on preprocessing steps applied

Missing values handled with mean imputation; invalid marks removed; categorical encoding; feature scaling.

Chapter 4: Development

Script details / Source code
System coded in Python: preprocessing, feature engineering, model training, evaluation, deployment.
Screen Shots / UI Design of simulation
Streamlit app allows input of student details, generates predictions, saves results.
Test reports
Logistic Regression: ~85%
SVM (RBF): ~87%
Random Forest: ~90% (best model).

Input :

Dataset ::

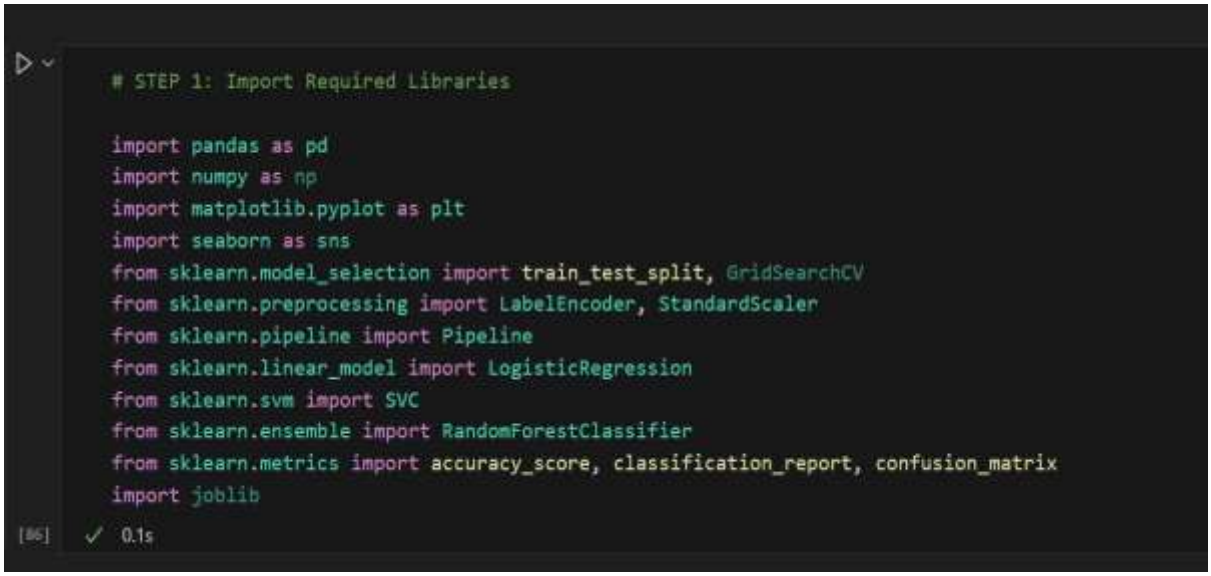
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Name	Age	Gender	Phone	EmailID	City	Location	Grade_10	Grade_12	JEE	CUET	Stream		
2	Ishaan Wil	17	Male	9.15E+11	ishaanwils	Bengaluru	Semi-Urba	56.22	57.7	148.16	668.84	B.Tech Civil		
3	Ishaan Joh	18	Male	9.14E+11	ishaanjohr	Sydney	Rural	93.64	92.23	187.66	645.15	B.Tech ECE		
4	Shaurya Br	20	Female	9.16E+11	shauryabr	Mumbai	Semi-Urba	54.18	64.77	174.53	718.74	B.Tech Mechanical		
5	Krishna Jol	18	Male	9.15E+11	krishnajoh	Dubai	Urban	63.81	58.2	146.53	492.62	B.Sc Data Science		
6	Atharv Sha	19	Male	9.19E+11	atharvshar	Bengaluru	Urban	82.44	74.42	283.08	555.21	B.Tech ECE		
7	Sai Miller	20	Male	9.15E+11	saimiller@	New York	Rural	90.65	95.36	101.19	762.74	B.Tech Mechanical		
8	Ishaan Wil	18	Female	9.13E+11	ishaanwils	Delhi	Rural	90.73	77.22532	260.44	719.07	B.Tech CS		
9	Ishaan Ver	18	Male	9.16E+11	ishaanverr	London	Semi-Urba	59.35	80.57	253.3	715.19	B.Tech CS		
10	Vivaan Joh	18	Female	9.13E+11	vivaanjohr	Ahmedaba	Rural	84.69		299.23	786.67	B.Tech CS		
11	Vihaan Sin	20	Male	9.14E+11	vihaansing	Ahmedaba	Urban	51.03	70.35	199.96	452.19	B.Sc Data Science		
12	Krishna Jol	20	Female	9.11E+11	krishnajoh	Bengaluru	Semi-Urba	86.96	64.65	100	100	BCA		
13	Krishna Pa	19	Male	9.11E+11	krishnapat	Ahmedaba	Rural	97.73	62.98	251.32	790.72	B.Tech CS		
14	Sai Verma	18	Female	9.16E+11	saiverma@	Toronto	Semi-Urban		100		100	BCA		
15	Aarav Khai	19	Male	9.12E+11	aaravkhan	Hyderabac	Rural	82.2	89.71	104.41	500.21	B.Sc Data Science		
16	Ishaan Sha	20	Female	9.10E+11	ishaanshar	Toronto	Semi-Urba	96.52	99.99	195.63	708.63	B.Tech CS		
17	Arjun Mille	18	Male	9.12E+11	arjunmiller	Hyderabac	Semi-Urba	50.83	69.99	114.6	454.03	B.Sc IT		
18	Vihaan Sin	20	Female	9.10E+11	vihaansing	Hyderabac	Semi-Urba	53.65			599.31	B.Tech Civil		
19	Shaurya Si	17	Female	9.18E+11	shauryasin	Dubai	Rural	100	67.2	100	100	BCA		
20	Atharv Sha	17	Male	9.10E+11	atharvshar	Hyderabac	Semi-Urba	69.36	77.22532	100	100	BCA		
21	Arjun Mille	19	Male	9.13E+11	arjunmiller	London	Rural	57.11	81.62	225.56	550.07	B.Tech Civil		
22	Atharv Ver	19	Female	9.10E+11	atharvverr	London	Urban	72.27	75.09	291.09	441.75	B.Tech Mechanical		
23	Arjun Pate	20	Female	9.11E+11	arjunpatel	New York	Semi-Urba	73.83	77.64	104.29	613.16	B.Tech Civil		
24	Aditya Bro	17	Male	9.13E+11	adityabrov	London	Urban	84.25	53.63	288.06	485.14	B.Tech Mechanical		
25	Vihaan Kh	18	Female	9.13E+11	vihaankha	Dubai	Semi-Urba	62.42	84.3		100	BCA		
26	Ishaan Sha	19	Male	9.13E+11	ishaanshar	London	Semi-Urba	88.3	54.12	172.01	594.75	B.Tech Civil		

Explanation:

This image shows a **student dataset** in tabular form (like Excel).
It contains details of students such as:

- **Personal info**: Name, Age, Gender, Phone, Email, City, Location
- **Academics**: Grade 10 marks, Grade 12 marks, JEE score, CUET score
- **Course/Stream**: e.g., B.Tech Civil, B.Tech CS, B.Sc Data Science, etc.

Basically, it's a collection of student records combining **demographic details, academic performance, and chosen stream**



```
# STEP 1: Import Required 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.preprocessing import LabelEncoder, StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import joblib
```

[86] ✓ 0.1s

Explaination:

This code is **Step 1: Importing libraries** needed for a machine learning project.

- **pandas, numpy** → data handling
- **matplotlib, seaborn** → visualization
- **scikit-learn (sklearn)** → preprocessing, model building (Logistic Regression, SVM, Random Forest), evaluation (accuracy, confusion matrix)
- **joblib** → model saving/loading

```
# STEP 2: Load Dataset

df = pd.read_csv("D:\\Sem 5\\Mini Project 2\\abc_final.csv")
print("Dataset Loaded Successfully")
print(df.head())
```

[20]

```
Dataset Loaded Successfully
              Name  Age  Gender         Phone                    EmailID  \
0    Ishaan Wilson   17    Male  9.153390e+11    ishaanwilson@gmail.com
1   Ishaan Johnson   18    Male  9.140990e+11   ishaanjohnson@gmail.com
2    Shaurya Brown   20  Female  9.161010e+11    shauryabrown@gmail.com
3  Krishna Johnson   18    Male  9.152570e+11  krishnajohnson@gmail.com
4    Atharv Sharma   19    Male  9.188130e+11    atharvsharma@gmail.com

        City    Location  Grade_10  Grade_12     JEE    CUET  \
0  Bengaluru  Semi-Urban     56.22     57.70  148.16  668.84
1     Sydney       Rural     93.64     92.23  187.66  645.15
2     Mumbai  Semi-Urban     54.18     64.77  174.53  718.74
3      Dubai       Urban     63.81     58.20  146.53  492.62
4  Bengaluru       Urban     82.44     74.42  283.08  555.21

              Stream
0       B.Tech Civil
1         B.Tech ECE
2  B.Tech Mechanical
3  B.Sc Data Science
```

Explaination:

This step loads the **dataset** (abc_final.csv) using pandas, confirms with a message, and shows the first 5 rows.

```python
# STEP 3: Data Cleaning

# Count missing values
print("\nMissing Values Before Cleaning:")
print(df.isnull().sum())

df = df[(df["Grade_10"] <= 100) & (df["Grade_12"] <= 100)]


# 3.1 - Handle missing values
def clean_missing_values(data):
    cleaned_df = data.copy()

    # Rows with only 1 missing numerical value → fill with column mean
    for col in ['Grade_10', 'Grade_12', 'JEE', 'CUET']:
        if col in cleaned_df.columns:
            cleaned_df[col] = cleaned_df[col].fillna(cleaned_df[col].mean())

    # Remove rows with more than 1 missing numerical value
    cleaned_df = cleaned_df.dropna(thresh=len(cleaned_df.columns) - 1)
    return cleaned_df

df = clean_missing_values(df)

print("\nMissing Values After Cleaning:")
print(df.isnull().sum())
```

[21]

```
Missing Values Before Cleaning:
Name           0
Age            0
Gender         0
Phone          0
EmailID        0
City           0
Location     133
Grade_10     303
Grade_12     317
JEE          378
CUET         364
Stream         0
dtype: int64

Missing Values After Cleaning:
Name           0
Age            0
Gender         0
Phone          0
EmailID        0
City           0
Location     119
Grade_10       0
...
JEE            0
CUET           0
Stream         0
```

Explaination:

This step **cleans data** by:

- Checking missing values
- Filling single missing scores with column mean
- Dropping rows with multiple missing values

```
# STEP 4:  Feature Engineering

df['JEE_100'] = df['JEE'] / 3.0
df['CUET_100'] = df['CUET'] / 8.0
df['Acad_Avg'] = (df['Grade_10'] + df['Grade_12']) / 2
df['Entrance_Avg'] = (df['JEE_100'] + df['CUET_100']) / 2
df['Combined_Score'] = 0.5 * df['Acad_Avg'] + 0.5 * df['Entrance_Avg']

print("\n Feature Engineering Complete")
print(df.head())

df["JEE_100"] = df["JEE_100"] * 2
df["CUET_100"] = df["CUET_100"] *2



X_temp = df.drop(columns=['Name', 'Phone', 'EmailID', 'Stream'])
y_temp = df['Stream']

# Encode categorical features in X_temp
X_temp_encoded = X_temp.copy()
for col in X_temp_encoded.columns:
    if X_temp_encoded[col].dtype == 'object':
        X_temp_encoded[col] = LabelEncoder().fit_transform(X_temp_encoded[col])

# Encode target
le = LabelEncoder()
y_temp_encoded = le.fit_transform(y_temp)
```

[22]

```
rf_temp = RandomForestClassifier(random_state=42)
rf_temp.fit(X_temp_encoded, y_temp_encoded)
importances = rf_temp.feature_importances_

feature_imp_df = pd.DataFrame({
    'Feature': X_temp_encoded.columns,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)

# Optional plot
plt.figure(figsize=(10,5))
plt.barh(feature_imp_df['Feature'], feature_imp_df['Importance'],color='pink',edgecolor='black')
plt.gca().invert_yaxis()
plt.xlabel("Importance Score")
plt.ylabel("Feature")
plt.title("Feature Importance (RandomForest)")
plt.show()



le = LabelEncoder()
y_encoded = le.fit_transform(df["Stream"])

X = df.drop(columns=["Name", "Phone", "EmailID", "Stream","City"])
y = df["Stream"]
```

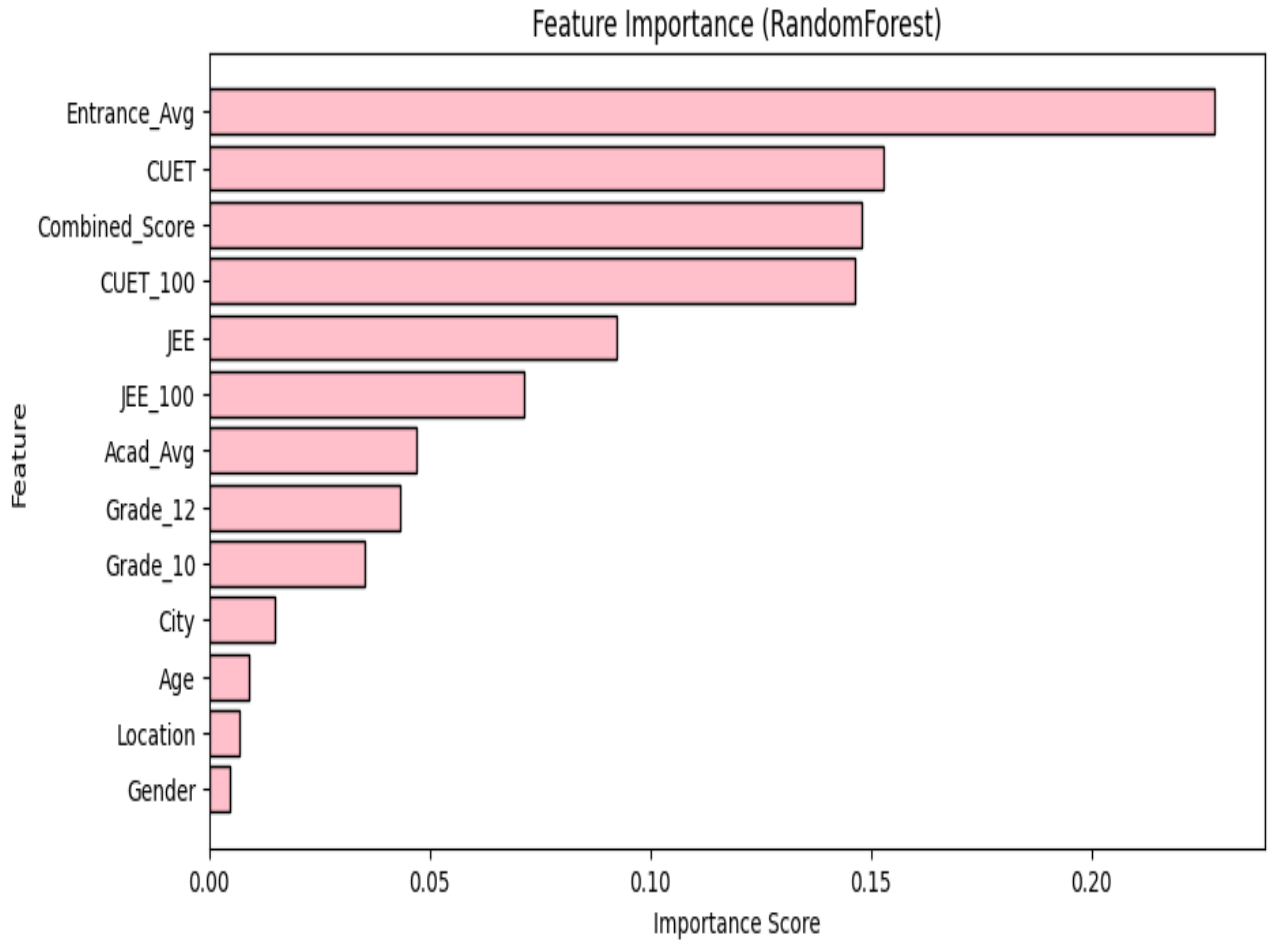
[22]

```
Feature Engineering Complete
              Name  Age  Gender         Phone                    EmailID  \
0    Ishaan Wilson   17    Male  9.153390e+11    ishaanwilson@gmail.com
1   Ishaan Johnson   18    Male  9.140990e+11   ishaanjohnson@gmail.com
2    Shaurya Brown   20  Female  9.161010e+11    shauryabrown@gmail.com
3  Krishna Johnson   18    Male  9.152570e+11  krishnajohnson@gmail.com
4    Atharv Sharma   19    Male  9.188130e+11    atharvsharma@gmail.com

        City    Location  Grade_10  Grade_12     JEE    CUET  \
0  Bengaluru  Semi-Urban     56.22     57.70  148.16  668.84
1     Sydney       Rural     93.64     92.23  187.66  645.15
2     Mumbai  Semi-Urban     54.18     64.77  174.53  718.74
3      Dubai       Urban     63.81     58.20  146.53  492.62
4  Bengaluru       Urban     82.44     74.42  283.08  555.21

              Stream    JEE_100  CUET_100  Acad_Avg  Entrance_Avg  \
0       B.Tech Civil  49.386667  83.60500    56.960     66.495833
1         B.Tech ECE  62.553333  80.64375    92.935     71.598542
2  B.Tech Mechanical  58.176667  89.84250    59.475     74.009583
3  B.Sc Data Science  48.843333  61.57750    61.005     55.210417
4         B.Tech ECE  94.360000  69.40125    78.430     81.880625

   Combined_Score
0       61.727917
1       82.266771
2       66.742292
3       58.107708
4       80.155313
```

Explaination:

This step is **Feature Engineering**.

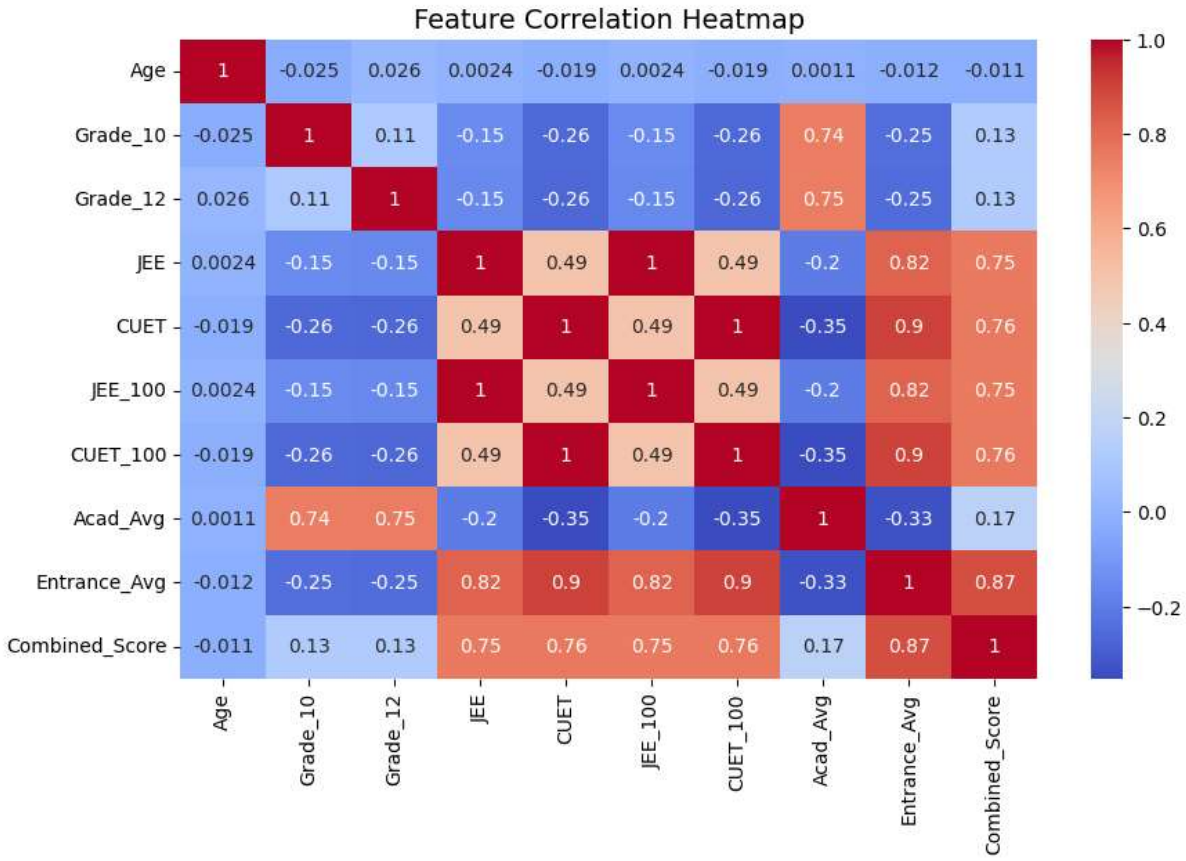
1. **Create new features**
    - Scale JEE & CUET to /100.
    - Compute averages: Academic Avg, Entrance Avg, Combined Score.
2. **Prepare data**
    - Drop unnecessary columns (Name, Phone, EmailID, Stream, City).
    - Encode categorical values with LabelEncoder.
    - Encode target (Stream).
3. **Feature Importance**
    - Train a RandomForestClassifier to check which features matter most.
    - Plot a bar chart of feature importance.




```
#Step 5 Ensure only numeric features are used for correlation
numeric_X = X.select_dtypes(include=['number'])

# Plot heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(numeric_X.corr(), annot=True, cmap="coolwarm")
plt.title("Feature Correlation Heatmap", fontsize=14)
plt.show()
```

Explaination:

This step selects only **numeric features** from the dataset and then plots a **heatmap of correlations** using seaborn.




```
# STEP 6: Train/Test Split

X = df.drop(columns=["Name", "Phone", "EmailID", "Stream","Age","Location","City","Gender","JEE","CUET"])
X = pd.get_dummies(X, drop_first=True)

# Handle missing values (fill with column mean)
X = X.fillna(X.mean())

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

[57] ✓ 0.0s

Explaination:

This step prepares data for modeling:

1. Drops unnecessary columns (Name, Phone, EmailID, Stream, etc.).
2. Converts categorical features into dummy variables (pd.get_dummies).
3. Fills any missing values with column mean.
4. Splits dataset into **train (80%)** and **test (20%)** using train_test_split.

```
# Step 7 Define models
models = {
    "Logistic Regression": Pipeline([
        ("scaler", StandardScaler()),
        ("clf", LogisticRegression(max_iter=2000, random_state=42))
    ]),
    "SVM (RBF Kernel)": Pipeline([
        ("scaler", StandardScaler()),
        ("clf", SVC(kernel="rbf", probability=True, random_state=42))
    ]),
    "Random Forest": Pipeline([
        ("clf", RandomForestClassifier(n_estimators=400, random_state=42))
    ])
}


accuracies = {}
predictions = {}

# 🔹 Train & evaluate
for name, pipe in models.items():
    pipe.fit(X_train, y_train)
    y_pred = pipe.predict(X_test)
    predictions[name] = y_pred
    acc = accuracy_score(y_test, y_pred)
    accuracies[name] = acc

    print(f"\n{name} Accuracy: {acc:.4f}")
```

```
        print(classification_report(y_test, y_pred, target_names=le.classes_))
        print("-" * 60)
    ✓ 2.5s

...
    Logistic Regression Accuracy: 0.8193
                       precision    recall  f1-score   support

    B.Sc Data Science       0.61      0.68      0.64        93
              B.Sc IT       0.71      0.32      0.44        38
             B.Tech CS      0.87      0.96      0.91       147
          B.Tech Civil      0.75      0.85      0.80       150
            B.Tech ECE      0.85      0.59      0.70       106
    B.Tech Mechanical       0.81      0.86      0.83       144
                  BCA       0.95      0.93      0.94       213

             accuracy                           0.82       891
            macro avg       0.79      0.74      0.75       891
         weighted avg       0.82      0.82      0.81       891

    ------------------------------------------------------------

    SVM (RBF Kernel) Accuracy: 0.8575
                       precision    recall  f1-score   support

    B.Sc Data Science       0.68      0.76      0.72        93
              B.Sc IT       0.82      0.61      0.70        38
             B.Tech CS      0.94      0.91      0.93       147
```

```
B.Sc Data Science       0.68      0.76      0.72        93
          B.Sc IT       0.82      0.61      0.70        38
         B.Tech CS      0.94      0.91      0.93       147
      B.Tech Civil      0.77      0.86      0.81       150
...
        macro avg       0.87      0.87      0.87       891
     weighted avg       0.89      0.89      0.89       891

------------------------------------------------------------
```

Explaination:

This step defines and tests ML models:

- Models used: **Logistic Regression, SVM (RBF), Random Forest** (inside pipelines with scaling where needed).
- Each model is trained on training data and tested on test data.
- Predictions and accuracies are stored and printed.

```
# Step 8 Confusion Matrix Heatmaps
for name, y_pred in predictions.items():
    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(6, 4))
    sns.heatmap(cm, annot=True, fmt="d", cmap="coolwarm")
    plt.title(f"Confusion Matrix - {name}")
    plt.ylabel("Actual")
    plt.xlabel("Predicted")
    plt.show()
```

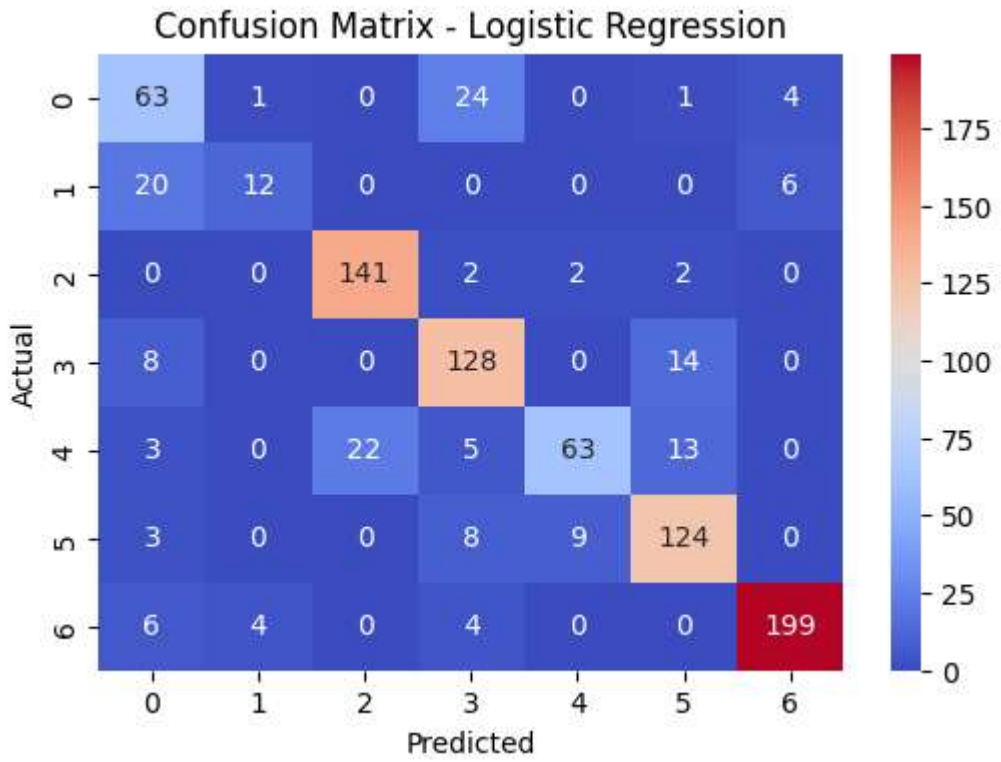
[59] ✓ 0.3s

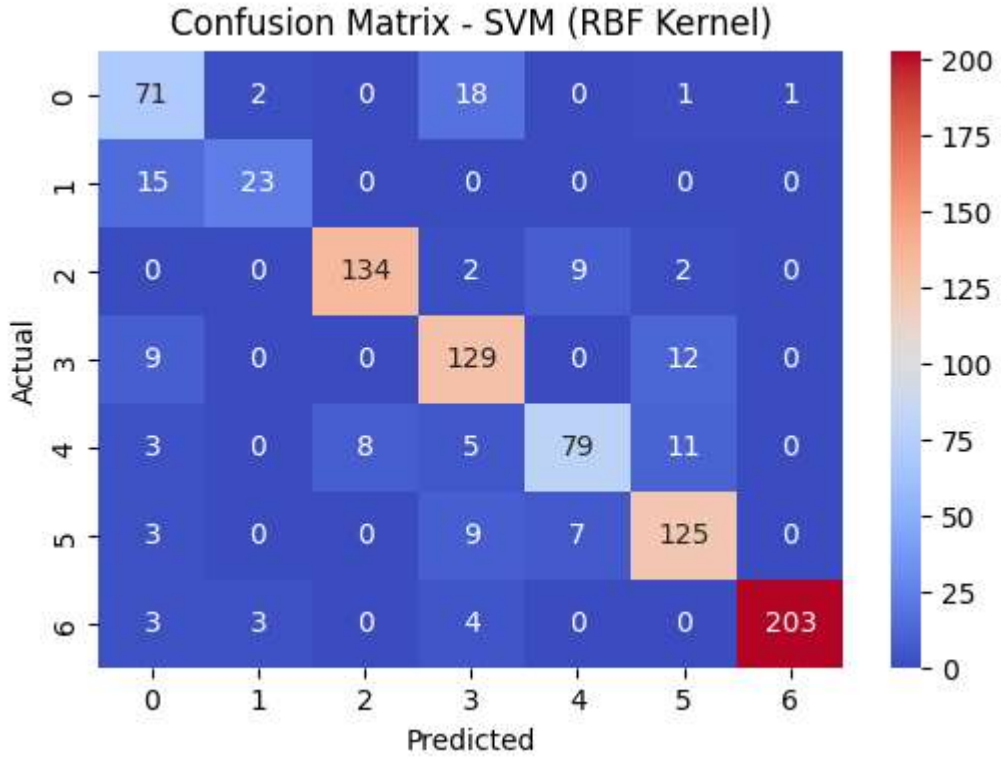
Explaination:

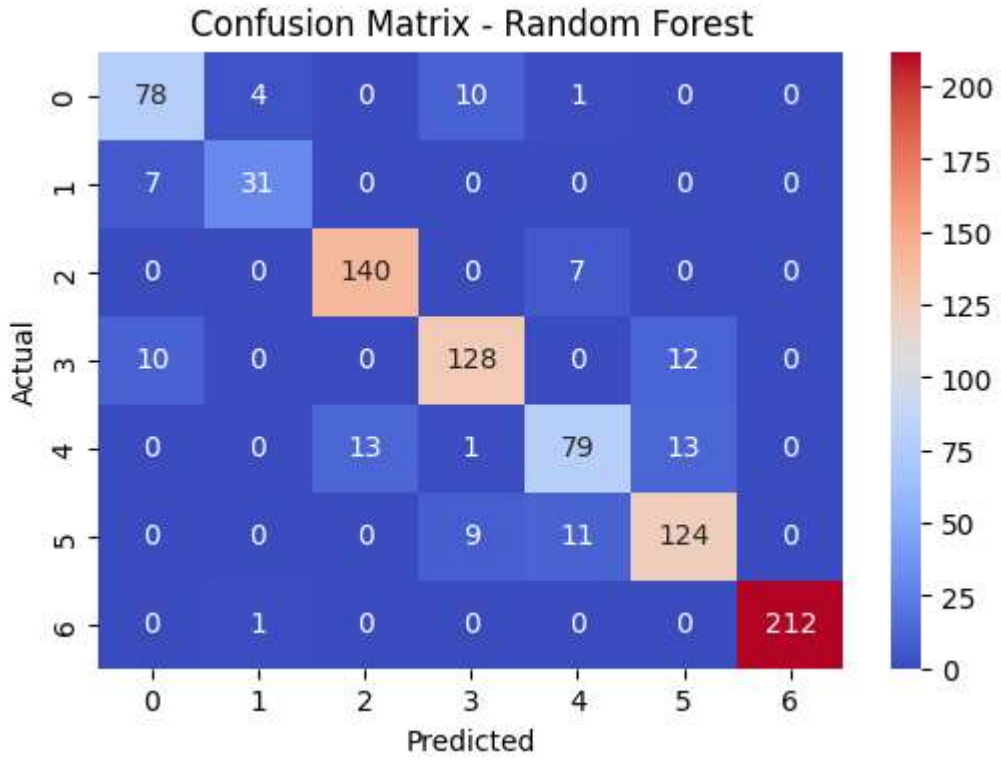
This code generates **confusion matrix heatmaps** for different model predictions.

Step-by-step:

1. **Loop through predictions** → Each model's predictions (y_pred) are taken from the predictions dictionary (name is the model name).
2. **Create confusion matrix** → confusion_matrix(y_test, y_pred) compares actual vs predicted values.
3. **Plot heatmap** → Using seaborn.heatmap() with annotations (annot=True) to show numbers inside the grid.
4. **Styling** →
    - figsize=(6,4) → set plot size.
    - cmap="coolwarm" → color scheme.
    - Labels (Actual, Predicted) and title (Confusion Matrix - model_name) added.
5. **Display** → plt.show() shows the heatmap.


Confusion Matrix - Logistic Regression
63 1 0 24 0 1 4
20 12 0 0 0 0 6
0 0 141 2 2 2 0
8 0 0 128 0 14 0
3 0 22 5 63 13 0
3 0 0 8 9 124 0
6 4 0 4 0 0 199
Actual
Predicted
0 1 2 3 4 5 6
175 150 125 100 75 50 25 0


Confusion Matrix - SVM (RBF Kernel)
71 2 0 18 0 1 1
15 23 0 0 0 0 0
0 0 134 2 9 2 0
9 0 0 129 0 12 0
3 0 8 5 79 11 0
3 0 0 9 7 125 0
3 3 0 4 0 0 203
Actual
Predicted
0 1 2 3 4 5 6
200 175 150 125 100 75 50 25 0




```
# STEP 9: Auto-pick Best Model & Save
best_model_name = max(accuracies, key=accuracies.get)
best_model = models[best_model_name]
best_model.fit(X, y)  # Retrain on full dataset

print(f"\n Best Model Selected: {best_model_name} (Accuracy: {accuracies[best_model_name] * 100:.2f}%)")

# Save model & label encoder
joblib.dump(best_model, "model.pkl")
joblib.dump(le, "label_encoder.pkl")
print(" Model & encoder saved successfully.")
```

[9] ✓ 3.1s

```
Best Model Selected: Random Forest (Accuracy: 88.89%)
Model & encoder saved successfully.
```

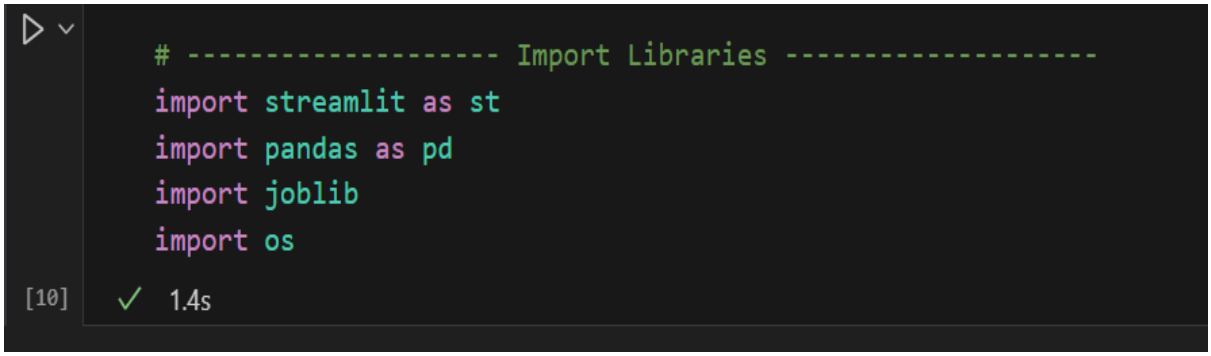
Explaination:

This code does the following in short:

1. **Auto-picks the best model** – From a dictionary of models and their accuracies, it selects the one with the highest accuracy.
2. best_model_name = max(accuracies, key=accuracies.get)

3. best_model = models[best_model_name]
4. **Retrains it** – Fits the chosen best model again on the full dataset (X, y).
5. **Displays result** – Prints which model was selected and its accuracy.
6. **Saves the model & encoder** – Saves the trained best model (model.pkl) and label encoder (label_encoder.pkl) using joblib.
7. Output in your case:

- Best model = **Random Forest** with **88.89% accuracy**
- Model & encoder saved successfully.



Explaination:

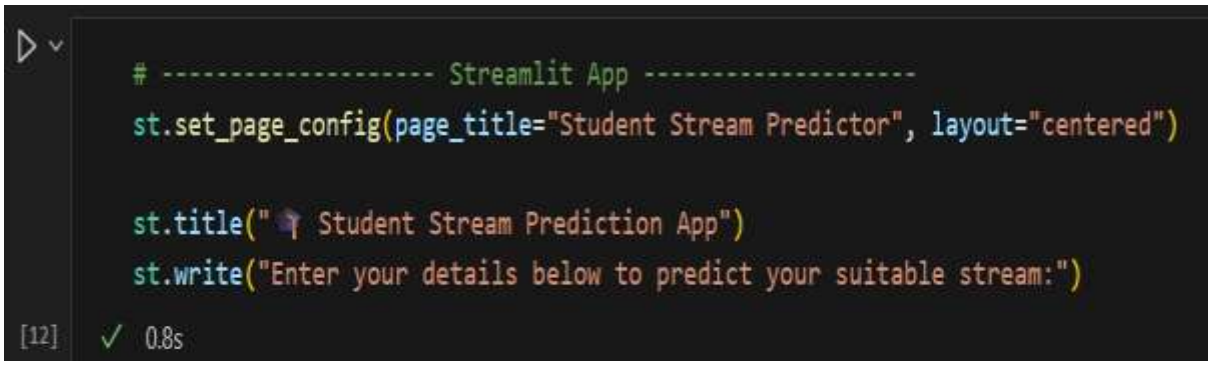
This code is simply **importing required libraries** for your project:

- streamlit as st → for creating a web app/dashboard.
- pandas as pd → for handling and analyzing data.
- joblib → for loading/saving trained ML models.
- os → for interacting with the operating system (like checking files, paths).

Explanation:

This code is **loading a pre-trained machine learning model** from a file:
m1 = joblib.load("model.pkl")

- joblib.load() → used to load saved Python objects (commonly ML models).
- "model.pkl" → the file where the trained model was stored.
- m1 → the loaded model object, ready to use for predictions.




```
# ------------------- Streamlit App --------------------
st.set_page_config(page_title="Student Stream Predictor", layout="centered")

st.title("🎓 Student Stream Prediction App")
st.write("Enter your details below to predict your suitable stream:")
```

Explanation:

This code is setting up a **Streamlit web app interface**:
st.set_page_config(page_title="Student Stream Predictor", layout="centered")
st.title("🎓 Student Stream Prediction App")
st.write("Enter your details below to predict your suitable stream:")

- st.set_page_config(...) → sets app settings (title shown on browser tab, layout).
- st.title(...) → displays the main title of the app.
- st.write(...) → adds descriptive text/instructions for users.

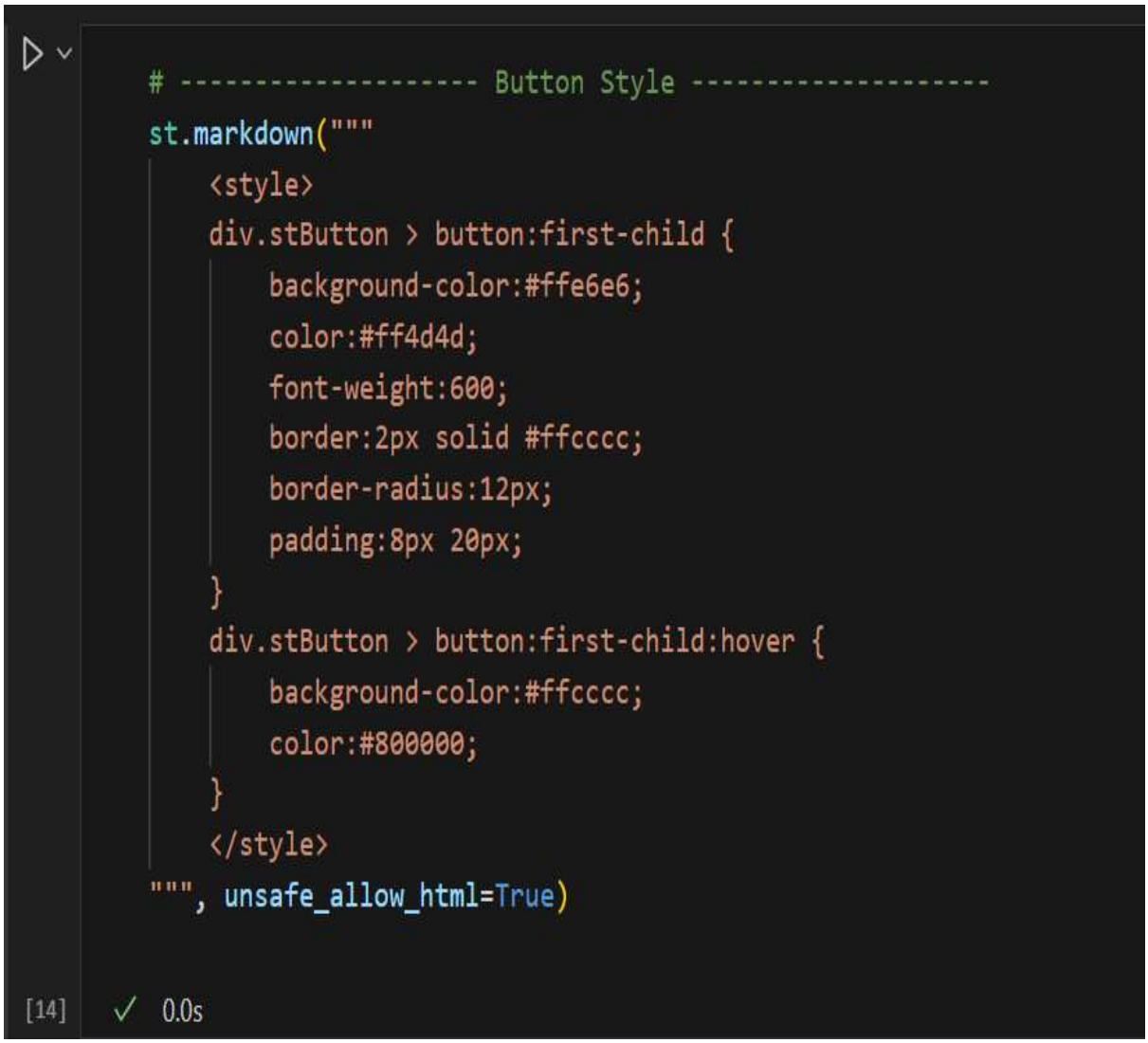
```python
# ------------------ User Inputs ------------------

# Student basic details
st.subheader("📝 Student Information")
name = st.text_input("Name")
age = st.number_input("Age", min_value=18, max_value=30, step=1)

# Layout in two columns
col1, col2 = st.columns(2)

with col1:
    gender = st.selectbox("Gender", ["Male", "Female", "Other"])
    grade_10 = st.number_input("Grade 10 Marks (%)", min_value=0, max_value=100, step=1)
    jee = st.number_input("JEE (out of 300)", min_value=0, max_value=300, step=1)

with col2:
    city = st.text_input("City")
    grade_12 = st.number_input("Grade 12 Marks (%)", min_value=0, max_value=100, step=1)
    cuet = st.number_input("CUET (out of 800)[Optional]", min_value=0, max_value=800, step=1)
```

[13] ✓ 0.0s

Explaination:

This code creates the **student input form** in Streamlit:

- st.subheader(" 📋 Student Information") → adds a section header.
- st.text_input, st.number_input, st.selectbox → collect inputs like **name, age, gender, city, marks, JEE, CUET scores**.
- st.columns(2) → arranges inputs neatly in **two side-by-side columns**.



```
# -------------------- Button Style --------------------
st.markdown("""
    <style>
    div.stButton > button:first-child {
        background-color:#ffe6e6;
        color:#ff4d4d;
        font-weight:600;
        border:2px solid #ffcccc;
        border-radius:12px;
        padding:8px 20px;
    }
    div.stButton > button:first-child:hover {
        background-color:#ffcccc;
        color:#800000;
    }
    </style>
""", unsafe_allow_html=True)
```

[14] ✓ 0.0s

Explaination:

This code customizes the **button style** in Streamlit using CSS inside st.markdown:

- Sets **background color, text color, border, padding, rounded corners** for buttons.
- Adds a **hover effect** → when the mouse moves over the button, the color changes.
- unsafe_allow_html=True → allows embedding custom HTML/CSS in Streamlit.

```python
# -------------------- Prediction --------------------
if st.button("🔮 Predict Stream"):
    # Step 1: Create DataFrame with raw inputs
    input_data = pd.DataFrame([[
        grade_10, grade_12, jee, cuet
    ]], columns=["Grade_10", "Grade_12", "JEE", "CUET"])

        # Step 2: Apply SAME feature engineering as training
    input_data["JEE_100"] = input_data["JEE"] / 3.0

    # Handle CUET as optional
    if cuet > 0:  # if CUET is entered
        input_data["CUET_100"] = input_data["CUET"] / 8.0
        input_data["Entrance_Avg"] = (input_data["JEE_100"] + input_data["CUET_100"]) / 2
    else:  # no CUET given → only JEE
        input_data["CUET_100"] = 0
        input_data["Entrance_Avg"] = input_data["JEE_100"]

    input_data["Acad_Avg"] = (input_data["Grade_10"] + input_data["Grade_12"]) / 2
    input_data["Combined_Score"] = 0.5 * input_data["Acad_Avg"] + 0.5 * input_data["Entrance_Avg"]

    # Step 3: Keep only trained features
    input_data = input_data[[
        "Grade_10", "Grade_12", "JEE_100", "CUET_100",
        "Acad_Avg", "Entrance_Avg", "Combined_Score"
    ]]
```

```python
    # Step 3: Keep only trained features
    input_data = input_data[["Grade_10", "Grade_12", "JEE_100", "CUET_100", "Acad_Avg", "Entrance_Avg", "Combined_Score"]]

    # Step 4: Predict
    pred_stream = ml.predict(input_data)[0]

    # Styled output
    st.markdown(
        f"""
        <div style="display:flex; justify-content:center; align-items:center; margin-top:20px;">
            <div style="
                background: radial-gradient(circle, #7c3aed, #5b21b6, #4c1d95);
                color: white;
                font-size: 22px;
                font-weight: bold;
                text-align: center;
                padding: 40px;
                border-radius: 50%;
                width: 220px;
                height: 220px;
                display: flex;
                justify-content: center;
                align-items: center;
                box-shadow: 0 0 25px rgba(124,58,237,1), 0 0 40px rgba(167,139,250,1);
            ">
                🎓 {pred_stream}
            </div>
        </div>
```

```
            </div>
        </div>
        """,
        unsafe_allow_html=True
    )
```

[15] ✓ 0.0s

Explanation:

This part does two things:

1. **Predict stream** → model (m1.predict) takes the processed input and outputs the student's suitable stream.
2. **Styled display** → shows the predicted stream inside a colorful, rounded, gradient-styled box using custom HTML/CSS.

```
# Extract engineered values
JEE_100 = input_data["JEE_100"].iloc[0]
CUET_100 = input_data["CUET_100"].iloc[0]
Acad_Avg = input_data["Acad_Avg"].iloc[0]
Entrance_Avg = input_data["Entrance_Avg"].iloc[0]
Combined_Score = input_data["Combined_Score"].iloc[0]
```

Explanation:

This code extracts the **calculated feature values** from the DataFrame (input_data) into separate variables:

- JEE_100, CUET_100 → normalized exam scores.
- Acad_Avg → average of Grade 10 & 12.
- Entrance_Avg → average of JEE & CUET.
- Combined_Score → overall score.

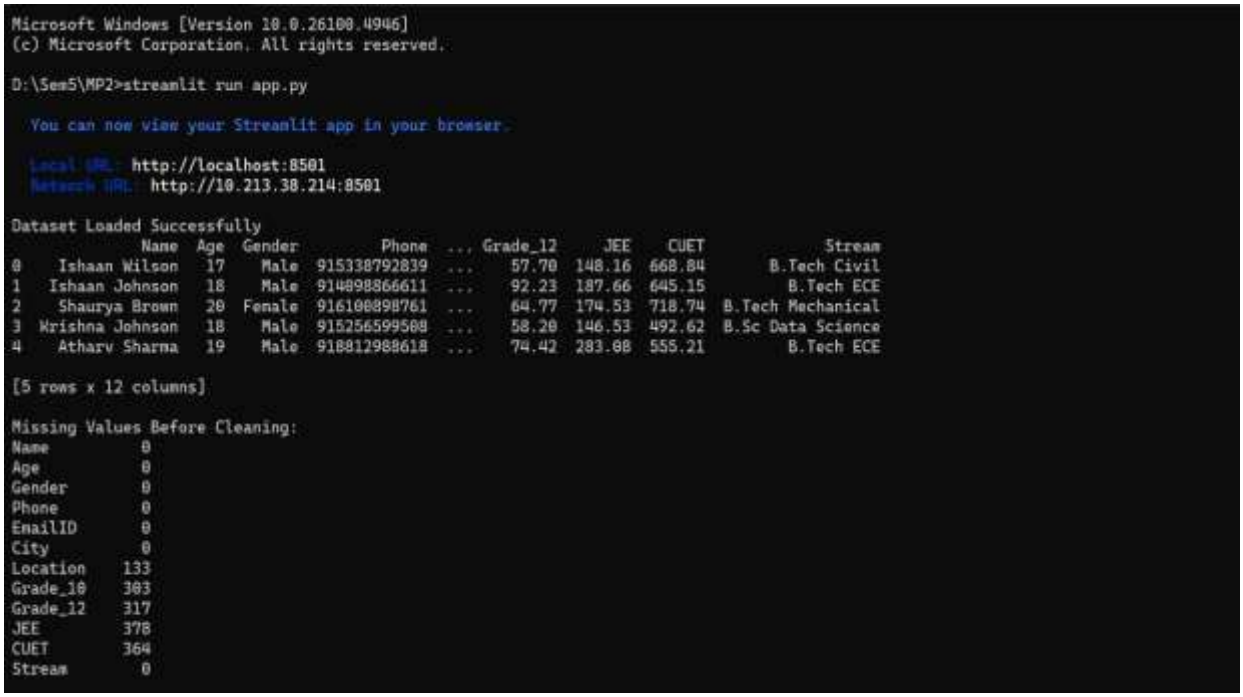
```
# Save input + prediction
save_data = pd.DataFrame([[name, age, grade_10, grade_12, JEE_100, CUET_100, Acad_Avg, Entrance_Avg, Combined_Score, gender, city, pred_stream]],
columns=[
        "Name", "Age", "Grade_10", "Grade_12", "JEE_100", "CUET_100",
        "Acad_Avg", "Entrance_Avg", "Combined_Score", "Gender", "City", "Predicted_Stream"])

if os.path.exists("predictions.csv"):
    save_data.to_csv("predictions.csv", mode="a", header=False, index=False)
else:
    save_data.to_csv("predictions.csv", index=False)

    st.info("📁 Data saved successfully!")
```

Explanation:

This code **saves the student's input and prediction** into a CSV file:

- Creates a DataFrame (save_data) with all details + predicted stream.
- If predictions.csv exists → appends new data.
- Else → creates a new CSV file.
- Shows message **"🗀 Data saved successfully!"**.

Output :



```
Microsoft Windows [Version 10.0.26100.4946]
(c) Microsoft Corporation. All rights reserved.

D:\Sem5\MP2>streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://10.213.38.214:8501

Dataset Loaded Successfully
             Name  Age  Gender         Phone  ... Grade_12     JEE    CUET            Stream
0    Ishaan Wilson   17    Male  915338792839  ...     57.70  148.16  668.84       B.Tech Civil
1   Ishaan Johnson   18    Male  914898866611  ...     92.23  187.66  645.15        B.Tech ECE
2    Shaurya Brown   20  Female  916100898761  ...     64.77  174.53  718.74  B.Tech Mechanical
3  Krishna Johnson   18    Male  915256599508  ...     58.20  146.53  492.62  B.Sc Data Science
4    Atharv Sharma   19    Male  918812988618  ...     74.42  283.08  555.21       B.Tech ECE

[5 rows x 12 columns]

Missing Values Before Cleaning:
Name          0
Age           0
Gender        0
Phone         0
EmailID       0
City          0
Location    133
Grade_10    303
Grade_12    317
JEE         378
CUET        364
Stream        0
```

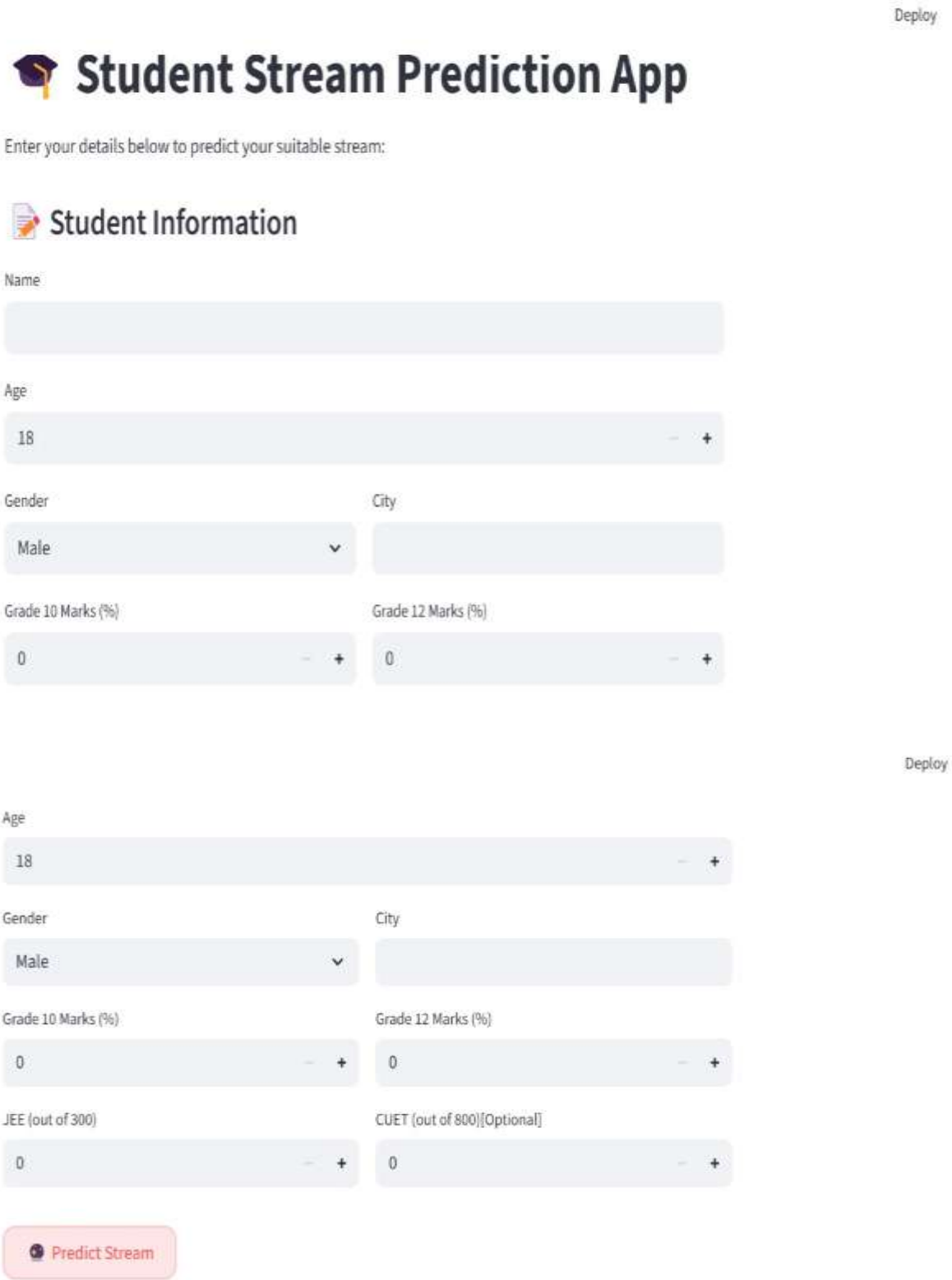
Explanation:

This screenshot shows a **Streamlit app running on localhost (port 8501)**.

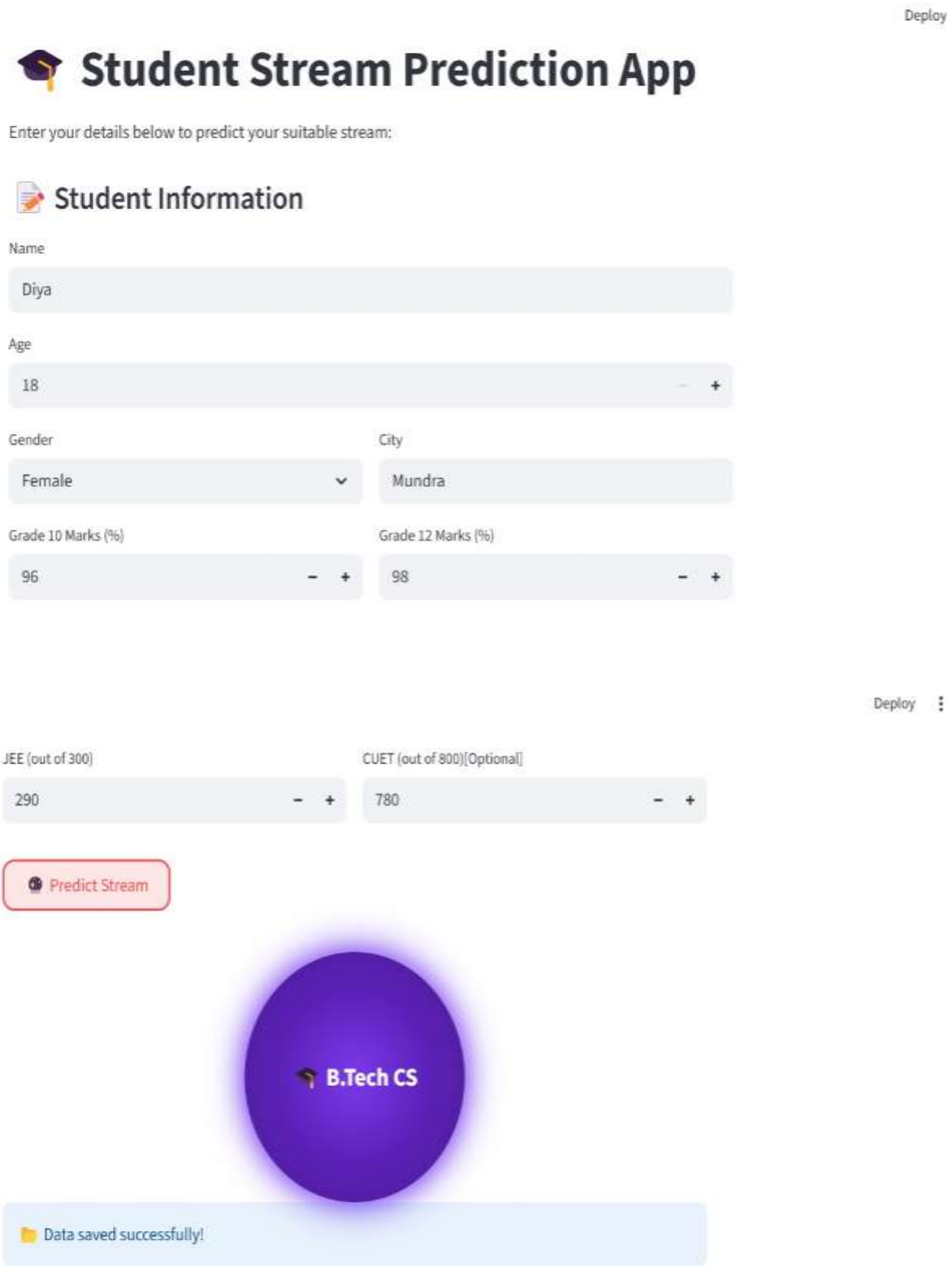
- The dataset is loaded successfully, displaying **student information** (Name, Age, Gender, Phone, Grade_12, JEE, CUET, Stream, etc.).
- It shows the **first 5 rows** of the dataset.

- Then it prints the **count of missing values** in each column:
  - Columns like Name, Age, Gender, Phone, Stream have **0 missing values**.
  - Columns like Location (133), Grade_10 (133), Grade_12 (317), JEE (378), CUET (364) have **missing values**.

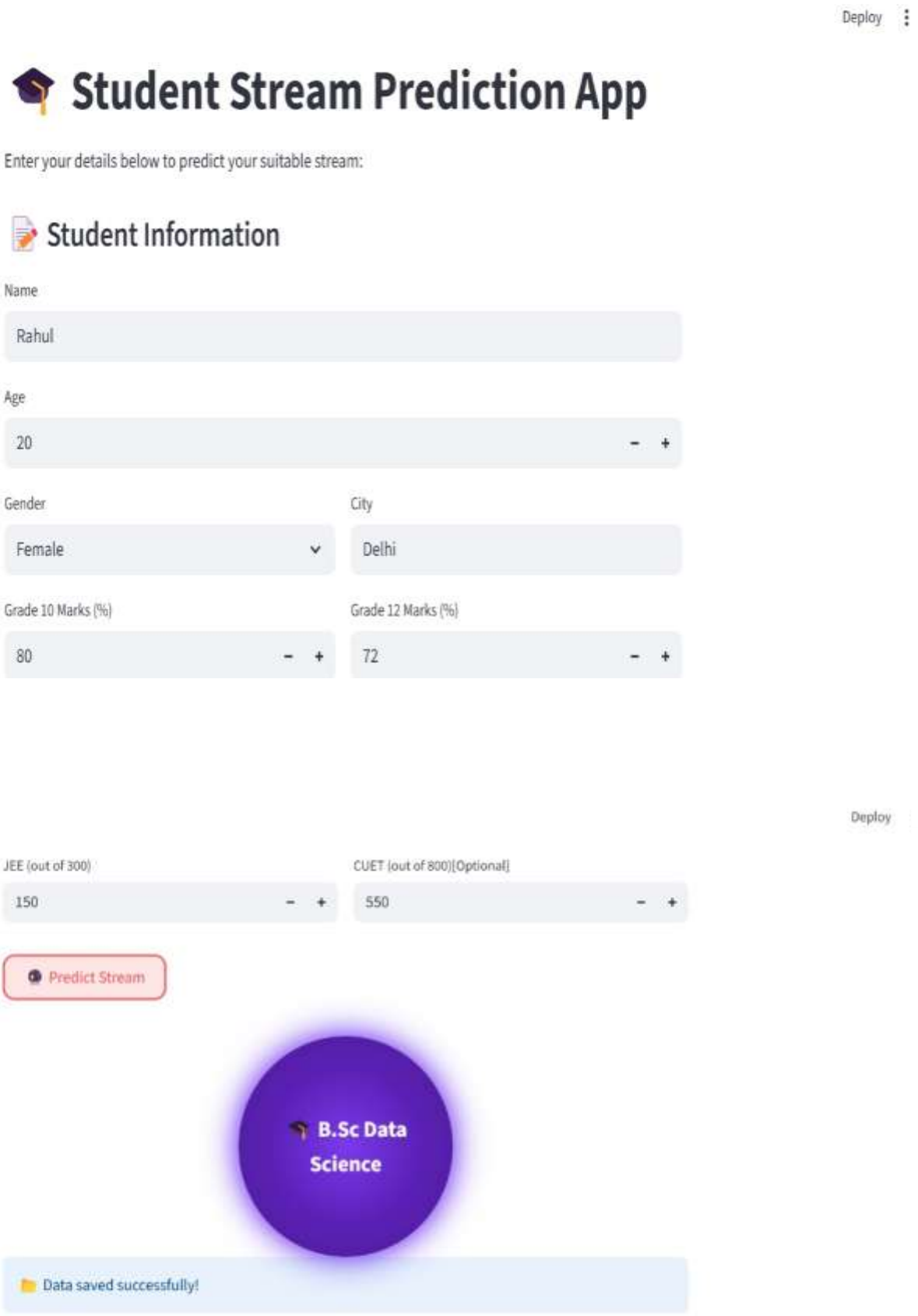
USER INTERFACE OUTPUT:



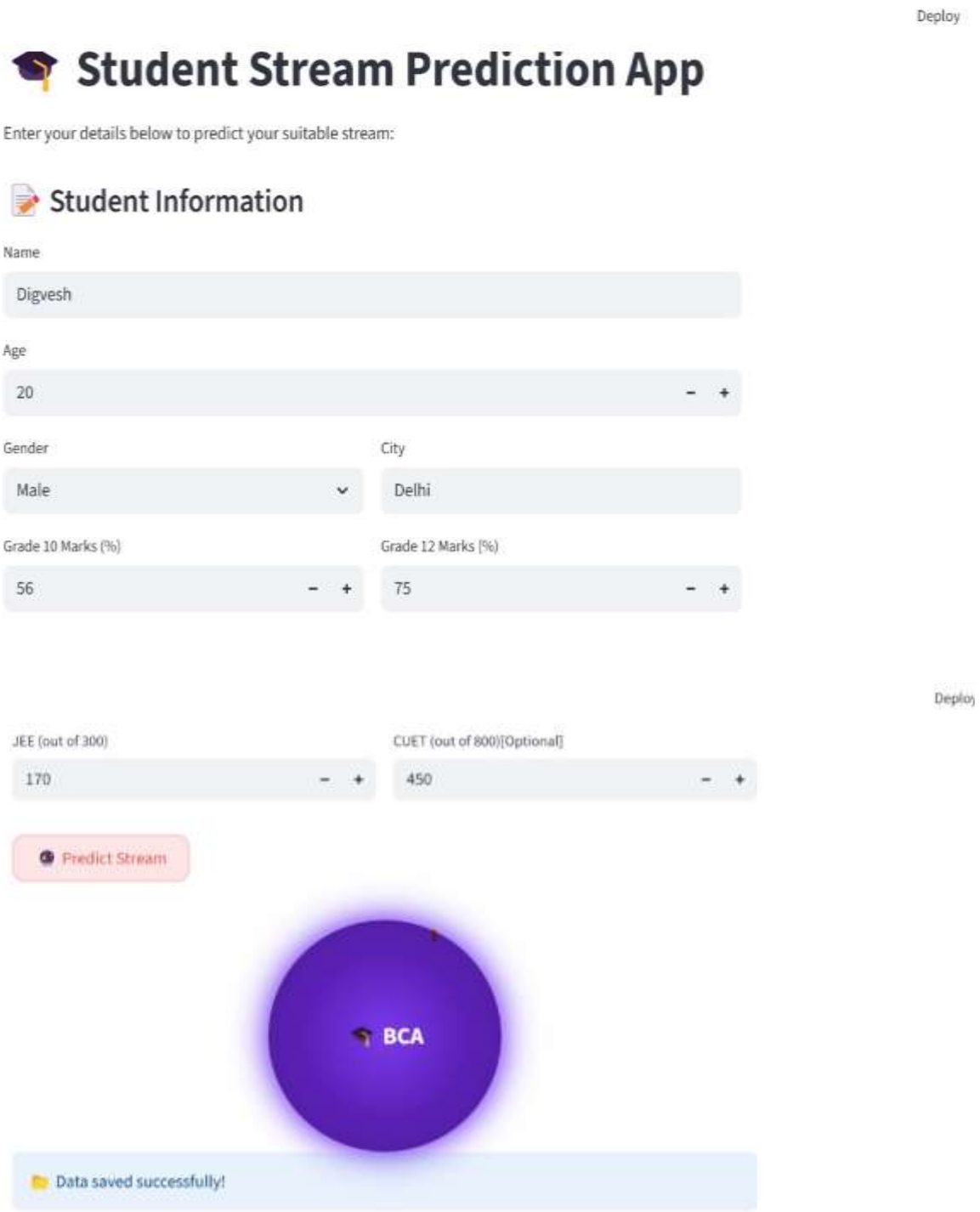
Sample 1:


Deploy
Student Stream Prediction App
Enter your details below to predict your suitable stream:
Student Information
Name
Diya
Age
18
Gender
Female
City
Mundra
Grade 10 Marks (%)
96
Grade 12 Marks (%)
98
Deploy
JEE (out of 300)
290
CUET (out of 800)[Optional]
780
Predict Stream
B.Tech CS
Data saved successfully!

Sample 2:


Deploy
Student Stream Prediction App
Enter your details below to predict your suitable stream:
Student Information
Name
Rahul
Age
20
Gender
Female
City
Delhi
Grade 10 Marks (%)
80
Grade 12 Marks (%)
72
Deploy
JEE (out of 300)
150
CUET (out of 800)[Optional]
550
Predict Stream
B.Sc Data Science
Data saved successfully!

Sample 3:


Deploy
Student Stream Prediction App
Enter your details below to predict your suitable stream:
Student Information
Name
Digvesh
Age
20
Gender
Male
City
Delhi
Grade 10 Marks (%)
56
Grade 12 Marks (%)
75
Deploy
JEE (out of 300)
170
CUET (out of 800)[Optional]
450
Predict Stream
BCA
Data saved successfully!

Dataset(Predictions Stored here):

File Edit View

Name,Age,Grade_10,Grade_12,JEE_100,CUET_100,Acad_Avg,Entrance_Avg,Combined_Score,Gender,City,Predicted_Stream
Diya,19,96,91,91.66666666666667,93.75,93.5,92.70833333333334,93.10416666666667,Female,Mundra,B.Tech CS
Divyesh,18,100,98,99.66666666666667,97.5,99.0,98.58333333333334,98.79166666666667,Female,,B.Tech CS
Keya,19,98,92,96.66666666666667,97.5,97.0,97.08333333333334,97.04166666666667,Female,Rajkot,B.Tech CS
Rahul,20,56,75,56.666666666666664,56.25,65.5,56.45833333333333,60.979166666666664,Male,Delhi,BCA

Chapter 5: Proposed Enhancements

Proposed Enhancements
Enhancements: larger dataset, deep learning, non-academic parameters, mobile deployment.

It suggests possible improvements for the system or project:

1. **Larger Dataset** – Using more data would improve accuracy, reliability, and generalization of the model.
2. **Deep Learning** – Applying advanced techniques (like neural networks) instead of only basic ML for better predictions.
3. **Non-Academic Parameters** – Including factors beyond academics (e.g., extracurriculars, personality traits, socio-economic background) for more holistic analysis.
4. **Mobile Deployment** – Making the system/app available on mobile platforms for easy accessibility and usability.

Chapter 6: Conclusion

Conclusion:

The project successfully demonstrates the use of **Machine Learning (ML) techniques** for predicting academic streams based on student performance data.

Various models were tested, and the **Random Forest classifier** proved to deliver the highest accuracy, highlighting its ability to handle complex datasets and reduce overfitting.

The system is further enhanced with a **Streamlit-based application**, ensuring user-friendliness and accessibility for both students and academic counselors. This interactive interface allows real-time predictions and recommendations, making it practical for educational institutions.

The study shows that integrating ML into academic decision-making can significantly aid students in selecting the most suitable stream, thereby reducing confusion and improving career guidance.

While the current system primarily relies on academic data, future improvements such as incorporating non-academic parameters, larger datasets, and deep learning methods could further enhance prediction accuracy and reliability.

In conclusion, the project not only demonstrates the feasibility of ML in education but also opens pathways for **smart, data-driven career counseling solutions** that are scalable, accessible, and impactful.

Chapter 7: Bibliography

Bibliography:

1.Scikit-learn Documentation – https://scikit-learn.org

2. Pandas Documentation – https://pandas.pydata.org

3. Streamlit Documentation – https://docs.streamlit.io

4. Seaborn Documentation – https://seaborn.pydata.org

5. Research papers on Student performance prediction.

6. Stack Overflow, GeeksforGeeks, W3Schools for code references.

Chapter 8 References

8.1 Offline References

[1] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2019.
[2] W. McKinney, *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and Jupyter*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2017.
[3] S. Raschka and V. Mirjalili, *Python Machine Learning*, 3rd ed. Birmingham, UK: Packt Publishing, 2019.

8.2 Online References

[4] Scikit-learn Developers, "Scikit-learn Documentation," [Online]. Available: https://scikit-learn.org. [Accessed: July. 29,2025].
[5] pandas Development Team, "pandas Documentation," [Online]. Available: https://pandas.pydata.org. [Accessed: Aug. 3, 2025].
[6] Streamlit Inc., "Streamlit Documentation," [Online]. Available: https://docs.streamlit.io. [Accessed: Aug. 4, 2025].
[7] Seaborn Developers, "Seaborn Documentation," [Online]. Available: https://seaborn.pydata.org. [Accessed: Aug. 5, 2025].
[8] Matplotlib Developers, "Matplotlib Documentation," [Online]. Available: https://matplotlib.org/stable/contents.html. [Accessed: Aug. 6, 2025].

[9] Stack Overflow, "Stack Overflow: Where Developers Learn, Share, & Build Careers," [Online]. Available: https://stackoverflow.com. [Accessed: Aug. 7, 2025].
[10] GeeksforGeeks, "GeeksforGeeks: A Computer Science Portal for Geeks," [Online]. Available: https://www.geeksforgeeks.org. [Accessed: Aug. 9, 2025].
[11] W3Schools, "W3Schools Online Web Tutorials," [Online]. Available: https://www.w3schools.com. [Accessed: Aug. 10, 2025].