PROTOTYPE

Submitted to,
Mr.Satheesh Kumar
Assistant Professor
PGDept of Computer Applications

Submitted by,
Joshuval James
23PMC131
2nd MCA

Exam Success Predictor

EduPredict is an innovative exam prediction system designed to transform how students and educators anticipate exam outcomes. By utilizing cutting-edge artificial intelligence and comprehensive data analytics, EduPredict offers accurate predictions on whether a student will pass or fail an exam, thereby providing valuable insights to guide study plans and interventions.

At the heart of EduPredict lies a sophisticated AI algorithm that evaluates a myriad of factors, including student performance history, study habits, engagement levels, and even external influences such as class participation and attendance. By integrating these data points, EduPredict delivers precise and personalized predictions, ensuring that each forecast is uniquely tailored to the student's academic journey.

1. Google Colab Script

The following link is based on the google colab script used for data preprocessing,model training and visualizations.

https://colab.research.google.com/drive/1OjX-T5TV0G8Je2sey1WouIc_MMpyxaZ3?usp=sharing

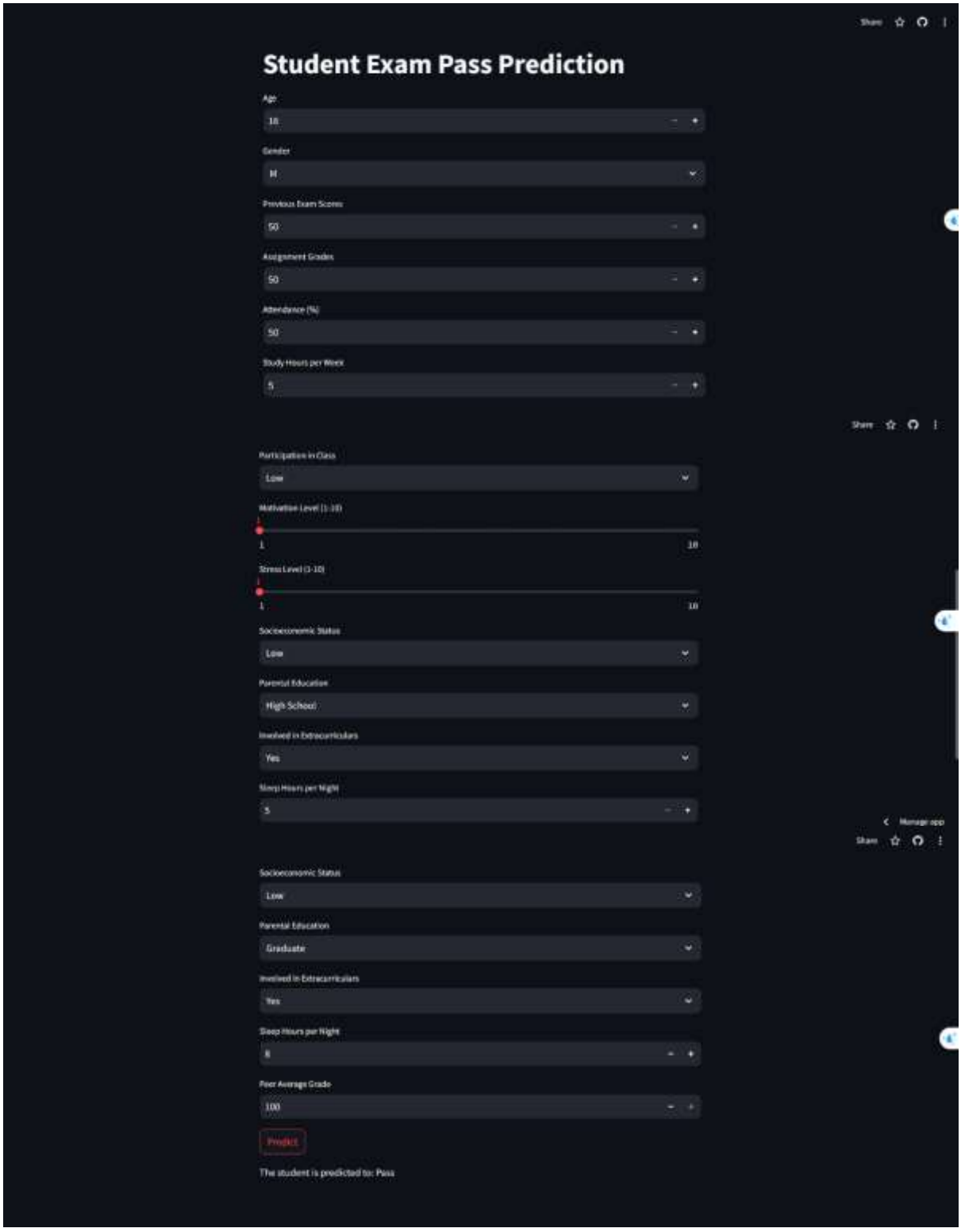
2. Web Application Prototype

2.1. Platform used:

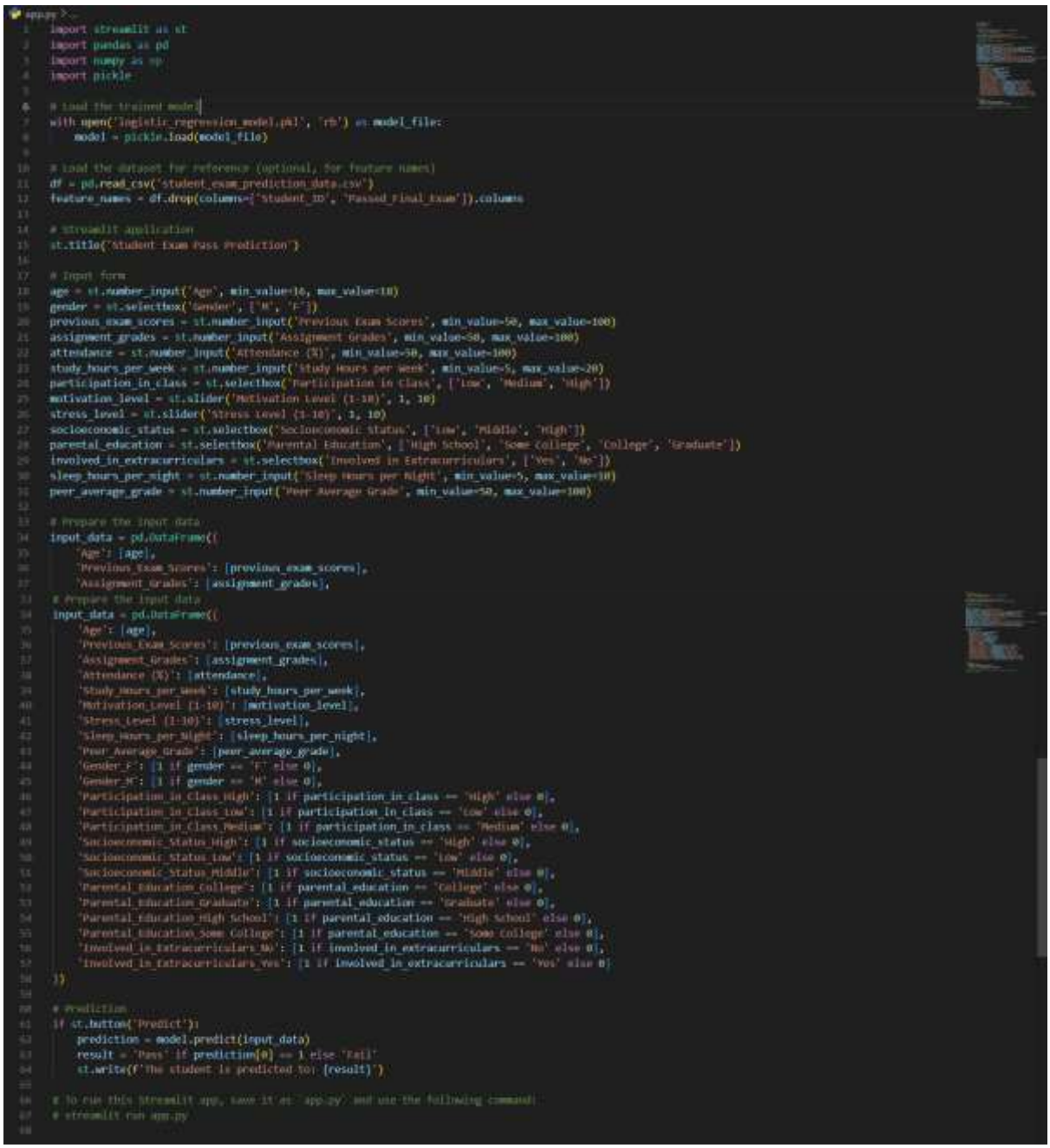
I have used at any application prototype interface and the link of deployed system is given below

https://appproject-g2wjsdjmrjrxotzgynwmwj.streamlit.app/

2.2. Screenshot of the application interface


Share
Student Exam Pass Prediction
Age
18
Gender
M
Previous Exam Scores
50
Assignment Grades
50
Attendance (%)
50
Study Hours per Week
5
Share
Participation in Class
Low
Motivation Level (1-10)
1
10
Stress Level (1-10)
1
10
Socioeconomic Status
Low
Parental Education
High School
Involved in Extracurriculars
Yes
Sleep Hours per Night
5
Manage app
Share
Socioeconomic Status
Low
Parental Education
Graduate
Involved in Extracurriculars
Yes
Sleep Hours per Night
8
Poor Average Grade
100
Predict
The student is predicted to: Pass

2.3. Python template or Source code for the application



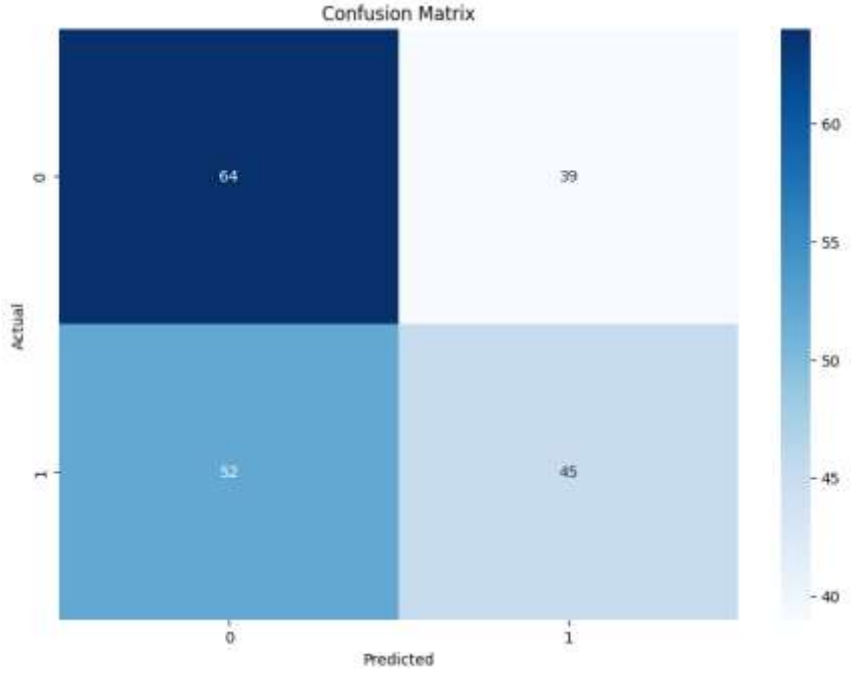
```
import streamlit as st
import pandas as pd
import numpy as np
import pickle

# Load the trained model
with open('logistic_regression_model.pkl', 'rb') as model_file:
    model = pickle.load(model_file)

# Load the dataset for reference (optional, for feature names)
df = pd.read_csv('student_exam_prediction_data.csv')
feature_names = df.drop(columns=['Student_ID', 'Passed_Final_Exam']).columns

# Streamlit application
st.title('Student Exam Pass Prediction')

# Input form
age = st.number_input('Age', min_value=16, max_value=18)
gender = st.selectbox('Gender', ['M', 'F'])
previous_exam_scores = st.number_input('Previous Exam Scores', min_value=50, max_value=100)
assignment_grades = st.number_input('Assignment Grades', min_value=50, max_value=100)
attendance = st.number_input('Attendance (%)', min_value=50, max_value=100)
study_hours_per_week = st.number_input('Study Hours per Week', min_value=5, max_value=20)
participation_in_class = st.selectbox('Participation in Class', ['Low', 'Medium', 'High'])
motivation_level = st.slider('Motivation Level (1-10)', 1, 10)
stress_level = st.slider('Stress Level (1-10)', 1, 10)
socioeconomic_status = st.selectbox('Socioeconomic Status', ['Low', 'Middle', 'High'])
parental_education = st.selectbox('Parental Education', ['High School', 'Some College', 'College', 'Graduate'])
involved_in_extracurriculars = st.selectbox('Involved in Extracurriculars', ['Yes', 'No'])
sleep_hours_per_night = st.number_input('Sleep Hours per Night', min_value=5, max_value=10)
peer_average_grade = st.number_input('Peer Average Grade', min_value=50, max_value=100)

# Prepare the input data
input_data = pd.DataFrame({
    'Age': [age],
    'Previous_Exam_Scores': [previous_exam_scores],
    'Assignment_Grades': [assignment_grades],
# Prepare the input data
input_data = pd.DataFrame({
    'Age': [age],
    'Previous_Exam_Scores': [previous_exam_scores],
    'Assignment_Grades': [assignment_grades],
    'Attendance (%)': [attendance],
    'Study_Hours_per_Week': [study_hours_per_week],
    'Motivation_Level (1-10)': [motivation_level],
    'Stress_Level (1-10)': [stress_level],
    'Sleep_Hours_per_Night': [sleep_hours_per_night],
    'Peer_Average_Grade': [peer_average_grade],
    'Gender_F': [1 if gender == 'F' else 0],
    'Gender_M': [1 if gender == 'M' else 0],
    'Participation_in_Class_High': [1 if participation_in_class == 'High' else 0],
    'Participation_in_Class_Low': [1 if participation_in_class == 'Low' else 0],
    'Participation_in_Class_Medium': [1 if participation_in_class == 'Medium' else 0],
    'Socioeconomic_Status_High': [1 if socioeconomic_status == 'High' else 0],
    'Socioeconomic_Status_Low': [1 if socioeconomic_status == 'Low' else 0],
    'Socioeconomic_Status_Middle': [1 if socioeconomic_status == 'Middle' else 0],
    'Parental_Education_College': [1 if parental_education == 'College' else 0],
    'Parental_Education_Graduate': [1 if parental_education == 'Graduate' else 0],
    'Parental_Education_High School': [1 if parental_education == 'High School' else 0],
    'Parental_Education_Some College': [1 if parental_education == 'Some College' else 0],
    'Involved_in_Extracurriculars_No': [1 if involved_in_extracurriculars == 'No' else 0],
    'Involved_in_Extracurriculars_Yes': [1 if involved_in_extracurriculars == 'Yes' else 0]
})

# Prediction
if st.button('Predict'):
    prediction = model.predict(input_data)
    result = 'Pass' if prediction[0] == 1 else 'Fail'
    st.write(f'The student is predicted to {result}')

# To run this Streamlit app, save it as `app.py` and use the following command:
# streamlit run app.py
```

```
input_data = pd.DataFrame({
    'Age': [age],
    'Previous_Exam_Scores': [previous_exam_scores],
    'Assignment_Grades': [assignment_grades],
    'Attendance (%)': [attendance],
    'Study_Hours_per_Week': [study_hours_per_week],
    'Motivation_Level (1-10)': [motivation_level],
    'Stress_Level (1-10)': [stress_level],
    'Sleep_Hours_per_Night': [sleep_hours_per_night],
    'Peer_Average_Grade': [peer_average_grade],
```

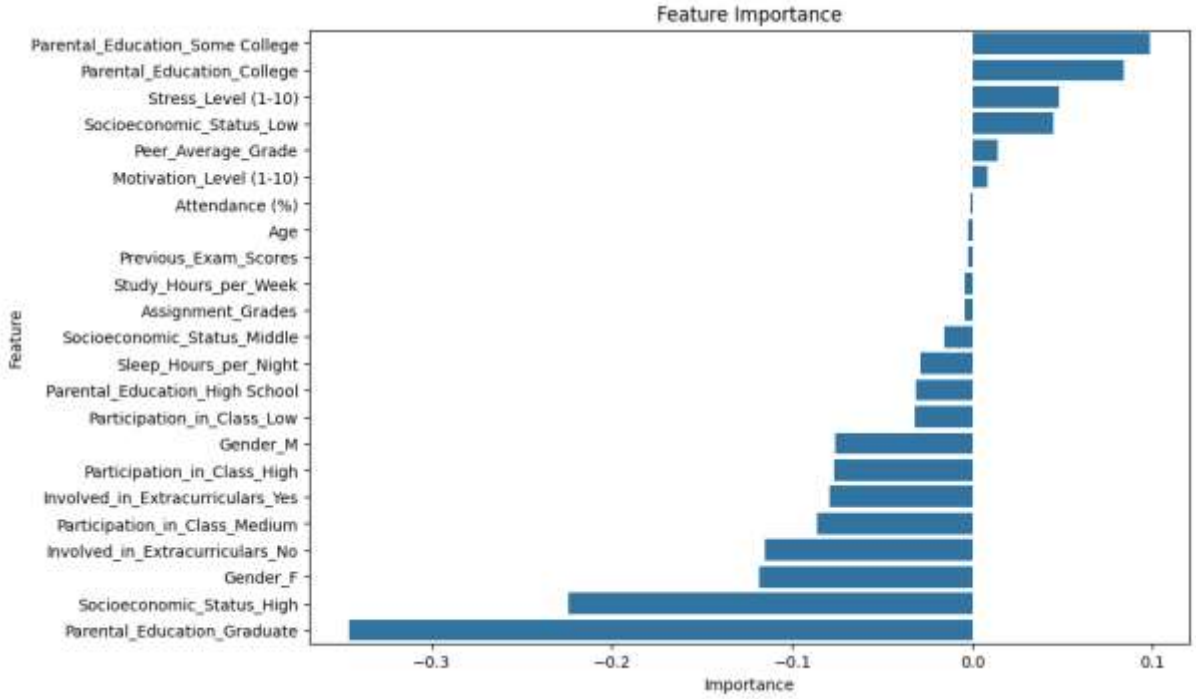
```
    'Gender_F': [1 if gender == 'F' else 0],
    'Gender_M': [1 if gender == 'M' else 0],
    'Participation_in_Class_High': [1 if participation_in_class == 'High' else 0],
    'Participation_in_Class_Low': [1 if participation_in_class == 'Low' else 0],
    'Participation_in_Class_Medium': [1 if participation_in_class == 'Medium' else 0],
    'Socioeconomic_Status_High': [1 if socioeconomic_status == 'High' else 0],
    'Socioeconomic_Status_Low': [1 if socioeconomic_status == 'Low' else 0],
    'Socioeconomic_Status_Middle': [1 if socioeconomic_status == 'Middle' else 0],
    'Parental_Education_College': [1 if parental_education == 'College' else 0],
    'Parental_Education_Graduate': [1 if parental_education == 'Graduate' else 0],
    'Parental_Education_High School': [1 if parental_education == 'High School' else 0],
    'Parental_Education_Some College': [1 if parental_education == 'Some College' else 0],
    'Involved_in_Extracurriculars_No': [1 if involved_in_extracurriculars == 'No' else 0],
    'Involved_in_Extracurriculars_Yes': [1 if involved_in_extracurriculars == 'Yes' else 0]
})
```

3. Visualization

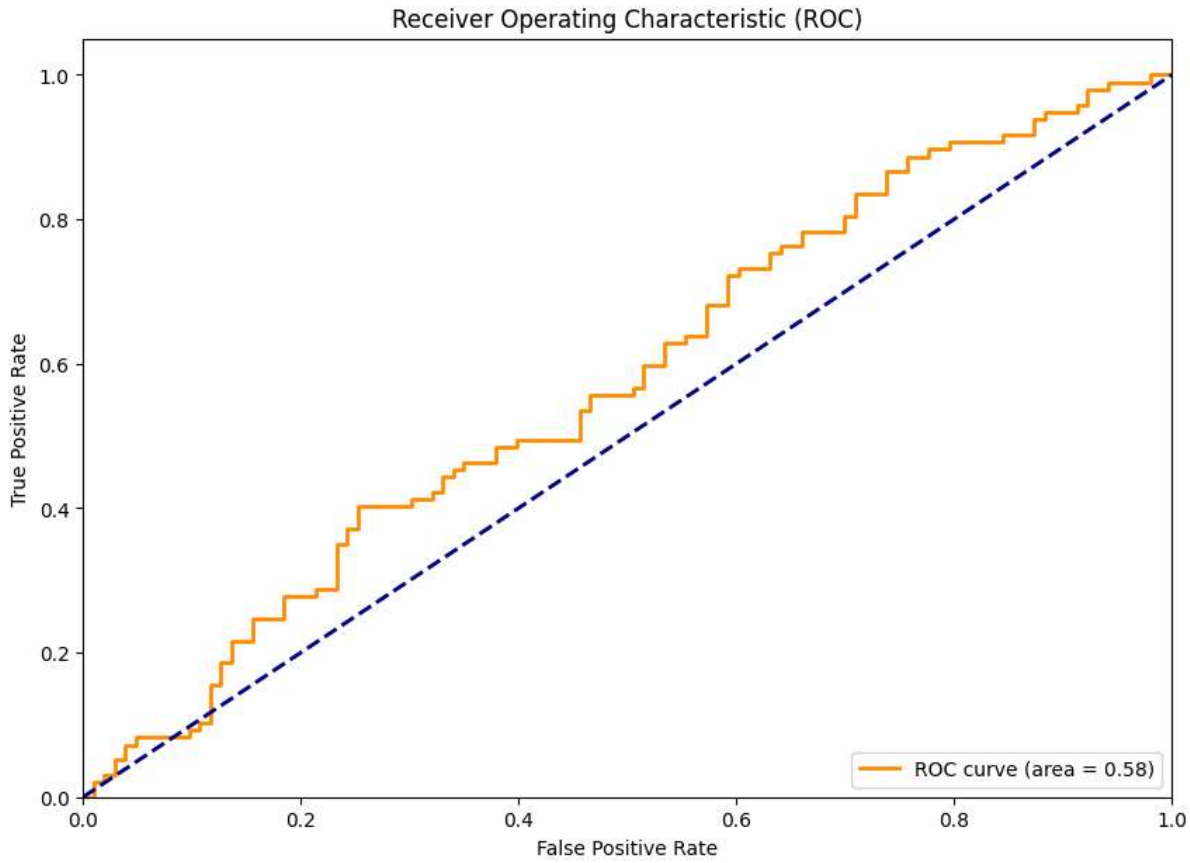
```
from sklearn.metrics import confusion_matrix
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(10, 7))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()
```



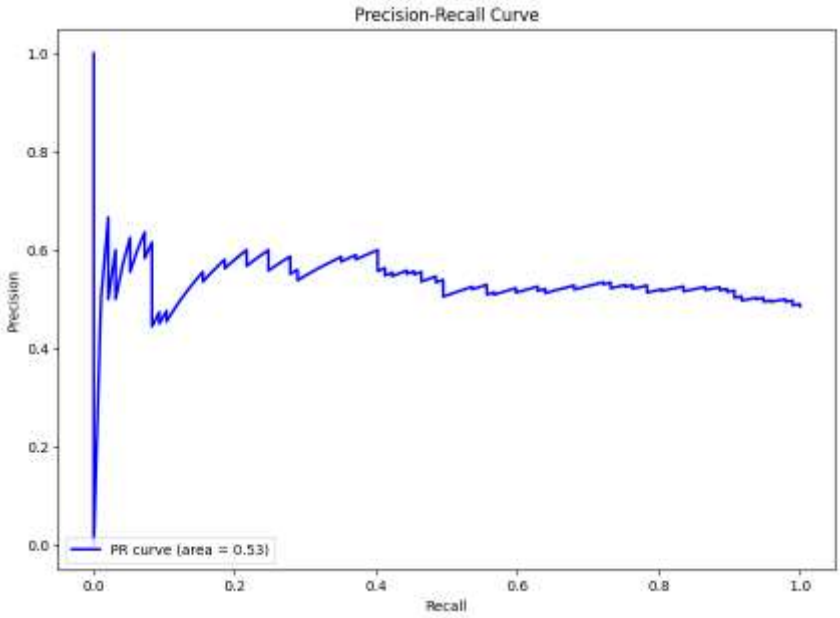
```
# Feature Importance
coefficients = model.coef_[0]
features = X.columns
importance = pd.DataFrame({'Feature': features, 'Importance': coefficients})
importance = importance.sort_values(by='Importance', ascending=False)

plt.figure(figsize=(10, 7))
sns.barplot(data=importance, x='Importance', y='Feature')
plt.title('Feature Importance')
plt.show()
```



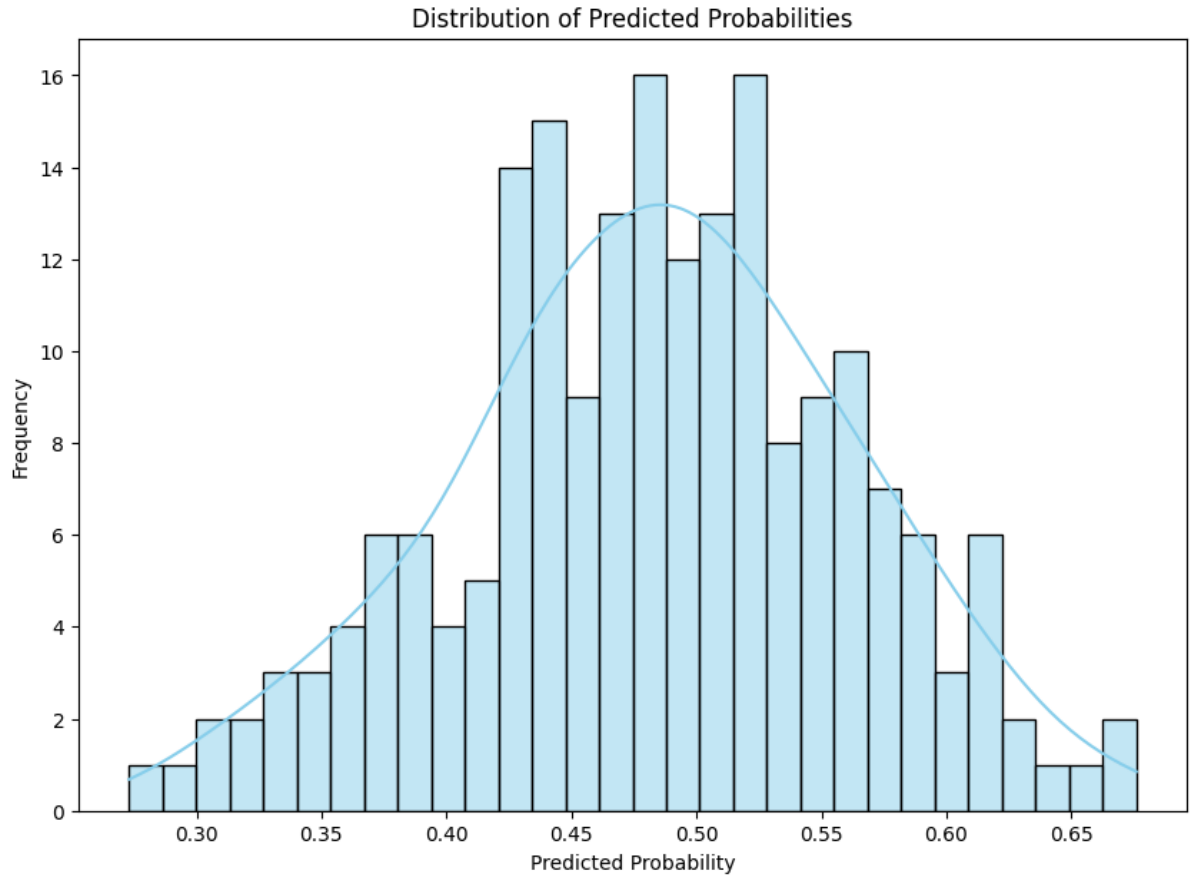

```
from sklearn.metrics import roc_curve, auc

# Compute ROC curve and ROC area
fpr, tpr, _ = roc_curve(y_test, model.predict_proba(X_test)[:,1])
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(10, 7))
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc='lower right')
plt.show()
```



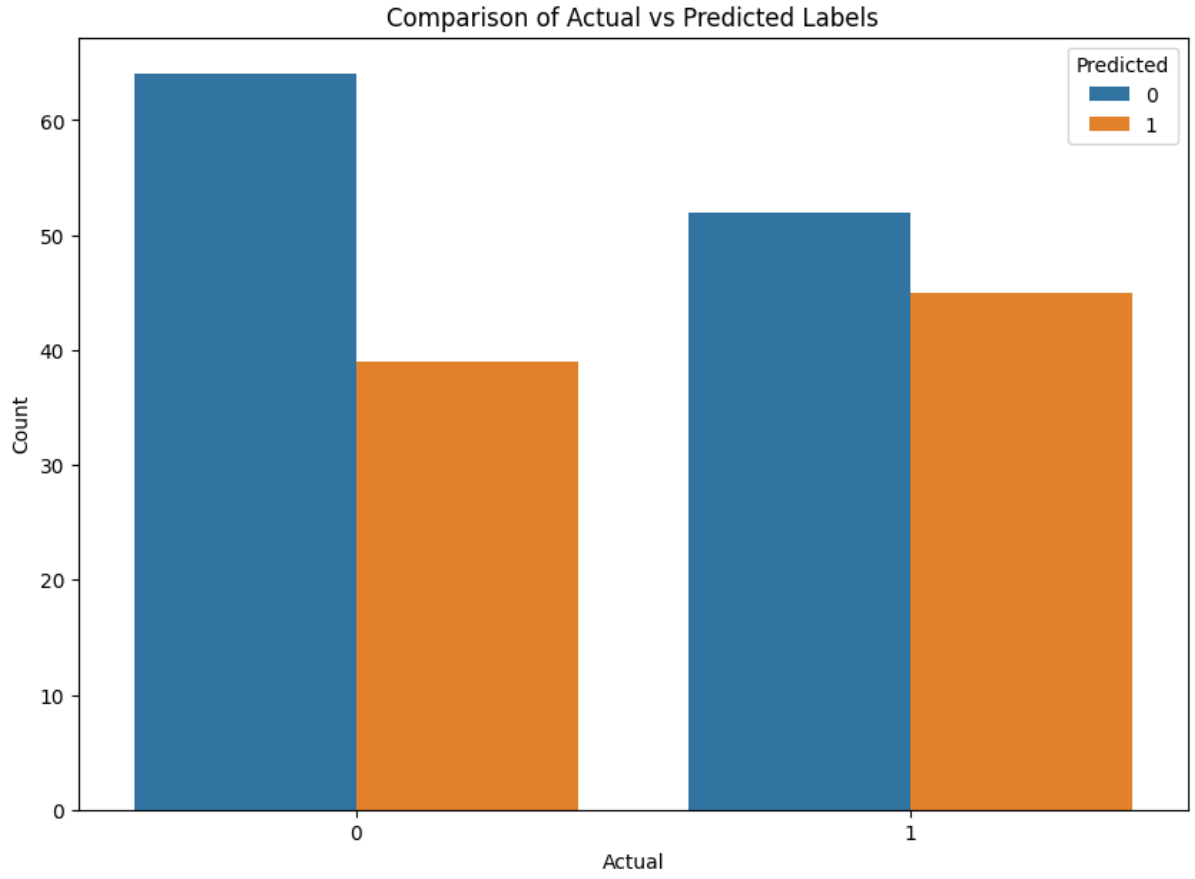

```
from sklearn.metrics import precision_recall_curve

# Compute Precision-Recall curve and PR area
precision, recall, _ = precision_recall_curve(y_test, model.predict_proba(X_test)[:,1])
pr_auc = auc(recall, precision)

plt.figure(figsize=(10, 7))
plt.plot(recall, precision, color='blue', lw=2, label='PR curve (area = %0.2f)' % pr_auc)
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.legend(loc='lower left')
plt.show()
```




```
y_prob = model.predict_proba(X_test)[:,1]

plt.figure(figsize=(10, 7))
sns.histplot(y_prob, kde=True, bins=30, color='skyblue')
plt.title('Distribution of Predicted Probabilities')
plt.xlabel('Predicted Probability')
plt.ylabel('Frequency')
plt.show()
```



```
results = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
plt.figure(figsize=(10, 7))
sns.countplot(data=results, x='Actual', hue='Predicted')
plt.title('Comparison of Actual vs Predicted Labels')
plt.xlabel('Actual')
plt.ylabel('Count')
plt.legend(title='Predicted')
plt.show()
```



CONCLUSION

The exam prediction system leverages machine learning to enhance student performance by predicting outcomes based on historical data. It offers several benefits, including early identification of at-risk students for timely intervention and personalized study plans, leading to improved student outcomes. Educators and administrators can make informed decisions regarding resource allocation and teaching strategies, supported by insights into performance trends. This system also reduces student stress by enabling realistic goal setting and providing proactive educational support. Additionally, the system is efficient and scalable, with automated predictions saving time and effort, and it can adapt to various educational settings. Continuous refinement of predictive models with ongoing data ensures enhanced accuracy and educational outcomes. In summary, the exam prediction

system is a powerful tool that improves student success through early intervention, personalized support, and data-driven insights.