Technical Report: Predicting Students' Exam Scores Using Regression Techniques

1. Project Overview

This machine learning project focuses on predicting students' exam scores using regression techniques. Input features include:

Hours Studied

Previous Exam Score

Attendance (%)

The target variable is the Final Exam Score. The goal is to build a regression model that predicts student performance and identifies key factors affecting academic outcomes.

2. Tools and Technologies

Programming Language: Python 3.x

Libraries:

NumPy, Pandas – data manipulation

Matplotlib, Seaborn – data visualization

Scikit-learn – machine learning models and evaluation

Jupyter Notebook / Google Colab – development environment

3. Dataset Description

A sample dataset may include:

Hours_Studied	Previous_Score	Attendance	Exam_Score (Target)
5.0	78	90	82
2.5	55	70	58
8.0	85	95	90

Hours_Studied: Continuous numeric variable.

Previous_Score: Previous test score (0–100 scale).

Attendance: Attendance percentage.

Exam_Score: Final exam score (Target variable).

4. Data Preprocessing

python

Copy

Edit

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Load dataset
```

```
df = pd.read_csv("student_scores.csv")


# Check for missing values

df.isnull().sum()


# Feature matrix (X) and target (y)

X = df[['Hours_Studied', 'Previous_Score', 'Attendance']]

y = df['Exam_Score']


# Train-test split

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


# Feature scaling

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)
```

5. Model Selection and Training

We evaluate multiple regression models:

5.1 Linear Regression

python

Copy

Edit

```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
```

lr.fit(X_train, y_train)

5.2 Ridge Regression

python

Copy

Edit

from sklearn.linear_model import Ridge

ridge = Ridge(alpha=1.0)

ridge.fit(X_train, y_train)

5.3 Random Forest Regressor

python

Copy

Edit

from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor(n_estimators=100, random_state=42)

rf.fit(X_train, y_train)

6. Model Evaluation

Evaluation metrics used:

Mean Absolute Error (MAE)

Mean Squared Error (MSE)

Root Mean Squared Error (RMSE)

R^2 Score

python

Copy

Edit

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np

def evaluate_model(model):
    y_pred = model.predict(X_test)
    print("MAE:", mean_absolute_error(y_test, y_pred))
    print("MSE:", mean_squared_error(y_test, y_pred))
    print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred)))
    print("R² Score:", r2_score(y_test, y_pred))

evaluate_model(lr)
evaluate_model(ridge)
evaluate_model(rf)
```

7. Feature Importance (for Tree-Based Models)

python

Copy

Edit

```
import matplotlib.pyplot as plt

importances = rf.feature_importances_
features = ['Hours_Studied', 'Previous_Score', 'Attendance']
```

```
plt.barh(features, importances)
plt.xlabel("Feature Importance")
plt.title("Random Forest Feature Importance")
plt.show()
```

8. Hyperparameter Tuning (Optional)

Using GridSearchCV for optimizing the Random Forest Regressor:

python

Copy

Edit

```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 5, 10],
    'min_samples_split': [2, 5],
}

grid = GridSearchCV(RandomForestRegressor(random_state=42), param_grid, cv=5, scoring='r2')
grid.fit(X_train, y_train)

print("Best Parameters:", grid.best_params_)
```

9. Deployment Options

Once the model is trained and tested, it can be deployed as:

Web App using Flask or FastAPI

Interactive dashboard using Streamlit

REST API for integration with school management systems

Example (Flask):

python

Copy

Edit

```
from flask import Flask, request, jsonify

app = Flask(__name__)


@app.route('/predict', methods=['POST'])

def predict():

    data = request.get_json()

    features = [[data['hours'], data['previous'], data['attendance']]]

    prediction = rf.predict(features)

    return jsonify({'predicted_score': prediction[0]})
```

10. Conclusion

This project demonstrates the effectiveness of regression algorithms in predicting student exam scores using study hours, previous scores, and attendance. Random Forest Regressor yielded the best performance in most scenarios. These models can be deployed in real-time systems to assist educators in early intervention and targeted academic support.

11. Future Work

Include more features like:

Sleep hours

Stress levels

Time management skills

Use deep learning models like ANN for better generalization

Develop mobile applications for student self-assessment