**Student grade prediction using linear and logistic regression and their comparison**

# 📦 1. Import Required Libraries

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from math import pi

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression, LogisticRegression

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score

# 📥 2. Load Dataset

df = pd.read\_csv("C:\\Users\\user\\OneDrive\\Documents\\Fundamentals of data science\\student-performance.csv") # Full path to your file

# ✅ Create a new binary target based on test\_score threshold

df['pass\_fail'] = (df['test\_score'] >= 50).astype(int)

# ✅ Show how many passed vs failed

print("🔍 New class distribution (after fixing):")

print(df['pass\_fail'].value\_counts())

# 🧼 3. Preprocessing: Encode Categorical Columns

df = df.apply(lambda col: col.astype('category').cat.codes if col.dtypes == 'object' else col)

# 🧾 4. Display Data Table and Column Names

print("🔍 Preprocessed Data Preview:")

print(df.head(10))

print("📑 Column Names:", df.columns.tolist())

# ✅ 5. Use 'pass\_fail' as Target

target\_col = 'pass\_fail'

X = df.drop(target\_col, axis=1)

y = df[target\_col]

# 📊 6. Correlation Heatmap

plt.figure(figsize=(12, 8))

sns.heatmap(df.corr(), annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)

plt.title("Feature Correlation Heatmap", fontsize=16)

plt.show()

print("✔ Full class distribution in pass\_fail:")

print(df['pass\_fail'].value\_counts())

# ✂️ 7. Train-Test Split (Stratified to preserve 0/1 balance)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.2, random\_state=42, stratify=y)

# 🤖 8. Train Models

# Linear Regression

lin\_model = LinearRegression()

lin\_model.fit(X\_train, y\_train)

lin\_preds = (lin\_model.predict(X\_test) >= 0.5).astype(int)

# Logistic Regression

log\_model = LogisticRegression(max\_iter=1000)

log\_model.fit(X\_train, y\_train)

log\_preds = log\_model.predict(X\_test)

# 🧪 9. Evaluation Function

def evaluate\_model(y\_true, y\_pred):

return {

"Accuracy": accuracy\_score(y\_true, y\_pred),

"Precision": precision\_score(y\_true, y\_pred),

"Recall": recall\_score(y\_true, y\_pred),

"F1 Score": f1\_score(y\_true, y\_pred)

}

# 📊 10. Model Performance

lin\_scores = evaluate\_model(y\_test, lin\_preds)

log\_scores = evaluate\_model(y\_test, log\_preds)

results\_df = pd.DataFrame([lin\_scores, log\_scores], index=['Linear Regression', 'Logistic Regression'])

print("\n📊 Model Performance Comparison:")

print(results\_df)

# 🌐 11. Radar Chart

def plot\_radar(data, title):

labels = list(data.columns)

num\_vars = len(labels)

angles = [n / float(num\_vars) \* 2 \* pi for n in range(num\_vars)]

angles += angles[:1]

fig, ax = plt.subplots(figsize=(8, 6), subplot\_kw=dict(polar=True))

for i, row in data.iterrows():

values = row.tolist()

values += values[:1]

ax.plot(angles, values, label=i, linewidth=2)

ax.fill(angles, values, alpha=0.25)

ax.set\_xticks(angles[:-1])

ax.set\_xticklabels(labels, fontsize=12)

ax.set\_yticklabels([])

ax.set\_title(title, size=16, pad=20)

ax.legend(loc='upper right', bbox\_to\_anchor=(1.3, 1.1))

plt.tight\_layout()

plt.show()

# 📈 12. Show Radar Chart

plot\_radar(results\_df, "Linear vs Logistic Regression - Model Performance")





