# Converted from PDF

Student exam perdiction  
Import necessary libraries  
Pandas:  
For loading, exploring, and manipulating structured data (like .csv files). Used for .read\_csv(),  
.head(), .describe(), etc.  
numpy  
Provides numerical computing support, especially for arrays and math functions. Often used  
under the hood by other libraries  
Seaborn  
Built on matplotlib; simplifies beautiful statistical plots (histograms, boxplots, heatmaps).  
Useful for Univariate/Bivariate Analysis  
matplotlib  
A plotting library for basic charts (line, bar, scatter). Used for understanding distributions and  
trends  
Data Collection and understanding  
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  
from sklearn.preprocessing import StandardScaler, LabelEncoder  
from sklearn.svm import SVC  
from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report  
import pickle  
import os # To check for file existence  
Data Collection and Understanding  
In [137…  
5/26/25, 4:16 PM  
basai2cporjectfile  
file:///C:/Users/HassanComputer/Downloads/basai2cporjectfile.html  
1/16

class DataLoader:  
 def \_\_init\_\_(self, filepath):  
 self.filepath = filepath  
   
 def describe(self):  
 return self.data.describe()  
 def info(self):  
 return self.data.info()  
 def head(self):  
 return self.data.head()  
 def tail(self):  
 return self.data.tail()  
 def load\_data(self):  
 self.data = pd.read\_csv(self.filepath)   
 return self.data  
   
 def show\_summary(self):  
 return self.data.describe()   
 def show\_info(self):  
 return self.data.info()  
Data Preprocessing & Analysis  
class DataPreprocessor:  
 def \_\_init\_\_(self, data):   
 self.data = data  
 def clean\_data(self):  
 self.data = self.data.dropna()   
 return self.data  
 def rename\_columns(self):  
 self.data = self.data.rename(columns={  
 'race/ethnicity': 'race\_ethnicity',  
 'parental level of education': 'parental\_education',  
 'test preparation course': 'prep\_course',  
 'math score': 'math\_score',  
 'reading score': 'reading\_score',  
 'writing score': 'writing\_score'  
 })  
 return self.data  
 def transform\_gender(self):  
 self.data['gender'] = self.data['gender'].map({'male': 'M', 'female': 'F'})  
 return self.data  
 def drop\_columns(self):  
 columns\_to\_drop = ['race\_ethnicity', 'parental\_education', 'lunch', 'prep\_c  
 self.data = self.data.drop(columns=columns\_to\_drop)  
In [138…  
In [139…  
5/26/25, 4:16 PM  
basai2cporjectfile  
file:///C:/Users/HassanComputer/Downloads/basai2cporjectfile.html  
2/16

return self.data  
 def mean(self):  
 return {  
 "math\_mean": self.data['math\_score'].mean(),  
 "reading\_mean": self.data['reading\_score'].mean(),  
 "writing\_mean": self.data['writing\_score'].mean()  
 }  
 def mode(self):  
 return {  
 "math\_mode": self.data['math\_score'].mode().values.tolist(),  
 "reading\_mode": self.data['reading\_score'].mode().values.tolist(),  
 "writing\_mode": self.data['writing\_score'].mode().values.tolist()  
 }  
 def median(self):  
 return {  
 "math\_median": self.data['math\_score'].median(),  
 "reading\_median": self.data['reading\_score'].median(),  
 "writing\_median": self.data['writing\_score'].median()  
 }  
Graph  
class Graph:  
 def \_\_init\_\_(self, df):  
 self.df = df  
 def plot\_histogram(self, bins=10):  
 plt.figure(figsize=(10, 6))  
 plt.hist(self.df['math\_score'], bins=bins, color='skyblue', edgecolor='blac  
 plt.title('Math Scores Distribution', fontsize=16)  
 plt.xlabel('Math Score', fontsize=14)  
 plt.ylabel('Number of Students', fontsize=14)  
 plt.grid(True, linestyle='--', alpha=0.7)  
 plt.xticks(fontsize=12)  
 plt.yticks(fontsize=12)  
 plt.tight\_layout()  
 plt.show()  
 def plot\_bar(self):  
 avg\_scores = self.df.groupby('gender')[['math\_score', 'reading\_score', 'wri  
 fig, axes = plt.subplots(1, 3, figsize=(18, 8))   
 axes[0].bar(avg\_scores.index, avg\_scores['math\_score'], color='skyblue')  
 axes[0].set\_title('Average Math Score by Gender')  
 axes[0].set\_ylabel('Score')  
 axes[0].set\_xlabel('Gender')  
 axes[1].bar(avg\_scores.index, avg\_scores['reading\_score'], color='lightgree  
 axes[1].set\_title('Average Reading Score by Gender')  
 axes[1].set\_ylabel('Score')  
 axes[1].set\_xlabel('Gender')  
In [140…  
5/26/25, 4:16 PM  
basai2cporjectfile  
file:///C:/Users/HassanComputer/Downloads/basai2cporjectfile.html  
3/16

axes[2].bar(avg\_scores.index, avg\_scores['writing\_score'], color='salmon')  
 axes[2].set\_title('Average Writing Score by Gender')  
 axes[2].set\_ylabel('Score')  
 axes[2].set\_xlabel('Gender')  
 plt.tight\_layout()  
 plt.show()  
 def plot\_pie(self):  
 gender\_counts = self.df['gender'].value\_counts()  
 plt.figure(figsize=(6, 6))  
 plt.pie(  
 gender\_counts,  
 labels=gender\_counts.index,  
 autopct='%1.1f%%',  
 startangle=90  
 )  
 plt.title('Gender Distribution')  
 plt.axis('equal')   
 plt.show()  
 def plot\_box(self):  
 plt.figure(figsize=(6, 5))  
 sns.boxplot(x='gender', y='reading\_score', data=self.df)  
 plt.title('Reading Score Distribution by Gender')  
 plt.xlabel('Gender')  
 plt.ylabel('Reading Score')  
 plt.show()  
 def plot\_scatter(self):  
 plt.figure(figsize=(6, 5))  
 sns.scatterplot(x='math\_score', y='reading\_score', hue='gender', data=self.  
 plt.title('Math vs Reading Scores by Gender')  
 plt.xlabel('Math Score')  
 plt.ylabel('Reading Score')  
 plt.legend(title='Gender')  
 plt.show()  
   
 def plot\_regression(self, x\_col, y\_col):  
 plt.figure(figsize=(8, 6))  
 sns.regplot(x=self.df[x\_col], y=self.df[y\_col])  
 plt.title(f'Regression Plot: {x\_col} vs {y\_col}')  
 plt.xlabel(x\_col)  
 plt.ylabel(y\_col)  
 plt.show()  
 def plot\_heatmap(self):  
 plt.figure(figsize=(10, 8))  
 sns.heatmap(self.df.corr(), annot=True, cmap="coolwarm", fmt=".2f")  
 plt.title("Correlation Heatmap")  
 plt.show()  
Univariate Analysis  
5/26/25, 4:16 PM  
basai2cporjectfile  
file:///C:/Users/HassanComputer/Downloads/basai2cporjectfile.html  
4/16

class Univariate(Graph):  
 def \_\_init\_\_(self, df):  
 super().\_\_init\_\_(df)  
 def plot\_histogram(self, column, bins=10):  
 plt.figure(figsize=(10, 6))  
 plt.hist(self.df[column], bins=bins, color='skyblue', edgecolor='black')  
 plt.title(f'Distribution of {column}', fontsize=16)  
 plt.xlabel(column, fontsize=14)  
 plt.ylabel('Frequency', fontsize=14)  
 plt.grid(True, linestyle='--', alpha=0.7)  
 plt.xticks(fontsize=12)  
 plt.yticks(fontsize=12)  
 plt.tight\_layout()  
 plt.show()  
 def plot\_box(self, column):  
 plt.figure(figsize=(6, 5))  
 sns.boxplot(y=self.df[column])  
 plt.title(f'Box Plot of {column}')  
 plt.ylabel(column)  
 plt.show()  
 def plot\_kde(self, column):  
 plt.figure(figsize=(8, 6))  
 sns.kdeplot(self.df[column], fill=True)  
 plt.title(f'Kernel Density Estimation of {column}')  
 plt.xlabel(column)  
 plt.ylabel('Density')  
 plt.show()  
Bivariate Analysis  
class Bivariate(Graph):  
 def \_\_init\_\_(self, df):  
 super().\_\_init\_\_(df)  
 def plot\_scatter(self, x\_col, y\_col, hue=None):  
 plt.figure(figsize=(8, 6))  
 sns.scatterplot(x=self.df[x\_col], y=self.df[y\_col], hue=self.df[hue] if hue  
 plt.title(f'Scatter Plot: {x\_col} vs {y\_col}')  
 plt.xlabel(x\_col)  
 plt.ylabel(y\_col)  
 plt.legend(title=hue)   
 plt.show()  
 def plot\_regression(self, x\_col, y\_col):  
 plt.figure(figsize=(8, 6))  
 sns.regplot(x=self.df[x\_col], y=self.df[y\_col])  
 plt.title(f'Regression Plot: {x\_col} vs {y\_col}')  
 plt.xlabel(x\_col)  
 plt.ylabel(y\_col)  
 plt.show()  
In [141…  
In [142…  
5/26/25, 4:16 PM  
basai2cporjectfile  
file:///C:/Users/HassanComputer/Downloads/basai2cporjectfile.html  
5/16

def plot\_heatmap(self):  
 plt.figure(figsize=(10, 8))  
 sns.heatmap(self.df.corr(), annot=True, cmap="coolwarm", fmt=".2f")  
 plt.title("Correlation Heatmap")  
 plt.show()  
Data Spilting  
class DataSplitter:  
   
 def \_\_init\_\_(self, X, y):  
   
 self.X = X  
 self.y = y  
 def split(self, test\_size=0.2, random\_state=42):  
   
 print(f"Splitting data with test\_size={test\_size}, random\_state={random\_sta  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split( self.X, self.y, test\_s  
 print(f"X\_train shape: {X\_train.shape}, X\_test shape: {X\_test.shape}")  
 print(f"y\_train shape: {y\_train.shape}, y\_test shape: {y\_test.shape}")  
 return X\_train, X\_test, y\_train, y\_test  
Model Training  
class ModelTrainer:  
 def \_\_init\_\_(self, kernel='linear'):  
 from sklearn.svm import SVC  
 self.model = SVC(kernel=kernel)  
 def train(self, X\_train, y\_train):  
 self.model.fit(X\_train, y\_train)  
 print("Model training complete.")  
 return self.model  
 def evaluate(self, X\_test, y\_test):  
 y\_pred = self.model.predict(X\_test)  
 acc = accuracy\_score(y\_test, y\_pred)  
 print(f"Model Accuracy: {acc:.4f}")  
 cm = confusion\_matrix(y\_test, y\_pred)  
 print("\nConfusion Matrix:")  
 print(cm)  
 # Plot confusion matrix  
 plt.figure(figsize=(6,5))  
 sns.heatmap(cm, annot=True, fmt="d", cmap='Blues', cbar=False)  
 plt.xlabel('Predicted')  
 plt.ylabel('Actual')  
 plt.title('Confusion Matrix')  
 plt.show()  
In [143…  
In [144…  
5/26/25, 4:16 PM  
basai2cporjectfile  
file:///C:/Users/HassanComputer/Downloads/basai2cporjectfile.html  
6/16

return acc, cm  
Model Storage  
class ModelStorage:  
 @staticmethod  
 def save\_model(model, filename='svm\_model.pkl'):  
 try:  
 with open(filename, 'wb') as f:  
 pickle.dump(model, f)  
 print(f"Model saved successfully to {filename}")  
 except Exception as e:  
 print(f"Error saving model: {e}")  
 @staticmethod  
 def load\_model(filename='svm\_model.pkl'):  
 if not os.path.exists(filename):  
 print(f"Error: Model file not found at {filename}")  
 return None  
 try:  
 with open(filename, 'rb') as f:  
 model = pickle.load(f)  
 print(f"Model loaded successfully from {filename}")  
 return model  
 except Exception as e:  
 print(f"Error loading model: {e}")  
 return None  
Main Excuetion  
if \_\_name\_\_ == "\_\_main\_\_":  
 data\_file\_path = "exams.csv"  
 try:  
 print("--- Loading Data ---")  
 loader = DataLoader(filepath=data\_file\_path)  
 raw\_data = loader.load\_data()  
 if raw\_data is None:  
 print("Data loading failed. Exiting.")  
 else:  
 print("Data loaded successfully.")  
   
 print("\n--- Preprocessing Data ---")  
 preprocessor = DataPreprocessor(raw\_data.copy())  
 preprocessor.rename\_columns()  
 preprocessor.transform\_gender()  
 processed\_data = preprocessor.data  
 if 'gender' in processed\_data.columns:  
 processed\_data['gender\_encoded'] = processed\_data['gender'].map({'M  
In [145…  
In [146…  
5/26/25, 4:16 PM  
basai2cporjectfile  
file:///C:/Users/HassanComputer/Downloads/basai2cporjectfile.html  
7/16

else:  
 raise KeyError("'gender' column not found in data.")  
 required\_cols = ['gender', 'math\_score', 'reading\_score', 'writing\_scor  
 if all(col in processed\_data.columns for col in required\_cols):  
 graph\_visualizer = Graph(df=processed\_data)  
 graph\_visualizer.plot\_histogram(bins=15)  
 graph\_visualizer.plot\_bar()  
 graph\_visualizer.plot\_pie()  
 graph\_visualizer.plot\_box()  
 graph\_visualizer.plot\_scatter()  
 univariate\_analyzer = Univariate(df=processed\_data)  
 univariate\_analyzer.plot\_histogram(column='reading\_score', bins=20)  
 univariate\_analyzer.plot\_box(column='writing\_score')  
 univariate\_analyzer.plot\_kde(column='math\_score')  
 bivariate\_analyzer = Bivariate(df=processed\_data)  
 bivariate\_analyzer.plot\_scatter(x\_col='math\_score', y\_col='writing\_  
 bivariate\_analyzer.plot\_regression(x\_col='reading\_score', y\_col='wr  
 numeric\_cols\_df = processed\_data.select\_dtypes(include=np.number)  
 if not numeric\_cols\_df.empty:  
 bivariate\_numeric\_analyzer = Bivariate(df=numeric\_cols\_df)  
 bivariate\_numeric\_analyzer.plot\_heatmap()  
 else:  
 print("Skipping visualization: some required columns missing.")  
 print("\n--- Starting Model Training ---")  
   
 X = processed\_data[['math\_score', 'reading\_score', 'writing\_score']]  
 y = processed\_data['gender\_encoded']  
   
 splitter = DataSplitter(X, y)  
 X\_train, X\_test, y\_train, y\_test = splitter.split(test\_size=0.2, random  
 scaler = StandardScaler()  
 X\_train\_scaled = scaler.fit\_transform(X\_train)  
 X\_test\_scaled = scaler.transform(X\_test)  
   
 trainer = ModelTrainer(kernel='linear')  
 model = trainer.train(X\_train\_scaled, y\_train)  
 print("\n--- Evaluating Model ---")  
 y\_pred = model.predict(X\_test\_scaled)  
 acc = accuracy\_score(y\_test, y\_pred)  
 print(f"Accuracy: {acc:.4f}")  
 print("\nConfusion Matrix:")  
 print(confusion\_matrix(y\_test, y\_pred))  
 print("\nClassification Report:")  
5/26/25, 4:16 PM  
basai2cporjectfile  
file:///C:/Users/HassanComputer/Downloads/basai2cporjectfile.html  
8/16

print(classification\_report(y\_test, y\_pred))  
 # Save model  
 ModelStorage.save\_model(model,"student\_exam\_svc\_model.pkl" )  
 print(f"\nModel saved to {"student\_exam\_svc\_model.pkl"}")  
 # Load and test model  
 loaded\_model = ModelStorage.load\_model("student\_exam\_svc\_model.pkl")  
 if loaded\_model:  
 sample\_preds = loaded\_model.predict(X\_test\_scaled[:5])  
 print(f"\nSample Predictions: {sample\_preds}")  
 print(f"Actual Values: {y\_test.iloc[:5].values}")  
 except FileNotFoundError:  
 print(f"File not found: {data\_file\_path}")  
 except pd.errors.EmptyDataError:  
 print(f"Data file is empty: {data\_file\_path}")  
 except KeyError as e:  
 print(f"Missing key in data: {e}")  
 except Exception as e:  
 print(f"Unexpected error: {e}")  
   
--- Loading Data ---  
Data loaded successfully.  
--- Preprocessing Data ---  
5/26/25, 4:16 PM  
basai2cporjectfile  
file:///C:/Users/HassanComputer/Downloads/basai2cporjectfile.html  
9/16

5/26/25, 4:16 PM  
basai2cporjectfile  
file:///C:/Users/HassanComputer/Downloads/basai2cporjectfile.html  
10/16

5/26/25, 4:16 PM  
basai2cporjectfile  
file:///C:/Users/HassanComputer/Downloads/basai2cporjectfile.html  
11/16

5/26/25, 4:16 PM  
basai2cporjectfile  
file:///C:/Users/HassanComputer/Downloads/basai2cporjectfile.html  
12/16

5/26/25, 4:16 PM  
basai2cporjectfile  
file:///C:/Users/HassanComputer/Downloads/basai2cporjectfile.html  
13/16

5/26/25, 4:16 PM  
basai2cporjectfile  
file:///C:/Users/HassanComputer/Downloads/basai2cporjectfile.html  
14/16

5/26/25, 4:16 PM  
basai2cporjectfile  
file:///C:/Users/HassanComputer/Downloads/basai2cporjectfile.html  
15/16

--- Starting Model Training ---  
Splitting data with test\_size=0.2, random\_state=42  
X\_train shape: (800, 3), X\_test shape: (200, 3)  
y\_train shape: (800,), y\_test shape: (200,)  
Model training complete.  
--- Evaluating Model ---  
Accuracy: 0.9100  
Confusion Matrix:  
[[93 12]  
 [ 6 89]]  
Classification Report:  
 precision recall f1-score support  
 0 0.94 0.89 0.91 105  
 1 0.88 0.94 0.91 95  
 accuracy 0.91 200  
 macro avg 0.91 0.91 0.91 200  
weighted avg 0.91 0.91 0.91 200  
Model saved successfully to student\_exam\_svc\_model.pkl  
Model saved to student\_exam\_svc\_model.pkl  
Model loaded successfully from student\_exam\_svc\_model.pkl  
Sample Predictions: [1 1 1 0 0]  
Actual Values: [0 0 1 0 0]  
   
In [ ]:  
5/26/25, 4:16 PM  
basai2cporjectfile  
file:///C:/Users/HassanComputer/Downloads/basai2cporjectfile.html  
16/16