Section 6: AI Tools & LLMs

Rohan Saha

April 10, 2025

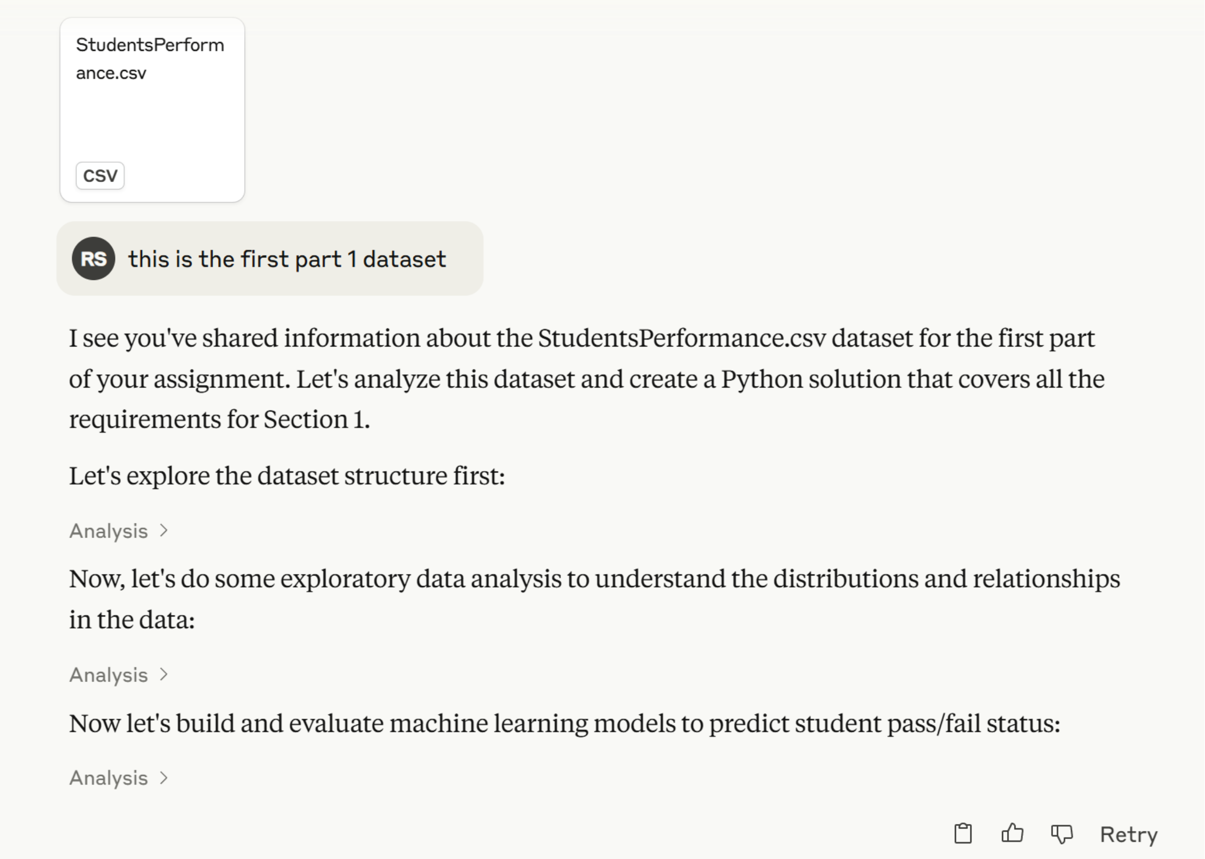
Option A: Prompt Engineering

I used Claude to help me with creating machine learning models for the student performance dataset. Below are the prompts I used and the responses received.

My Prompt

I shared the StudentsPerformance.csv dataset with Claude and asked for assistance with the Python/ML task:

see for task 1 I have to do this Tasks:
1. Load the dataset and perform data cleaning.
2. Perform EDA using visualizations (use seaborn, matplotlib).
3. Predict whether a student will pass or fail based on scores and other features.
4. Use Logistic Regression or Random Forest for classification.
5. Evaluate the model using accuracy, confusion matrix, F1-score.
6. Add comments/markdown to explain the steps.



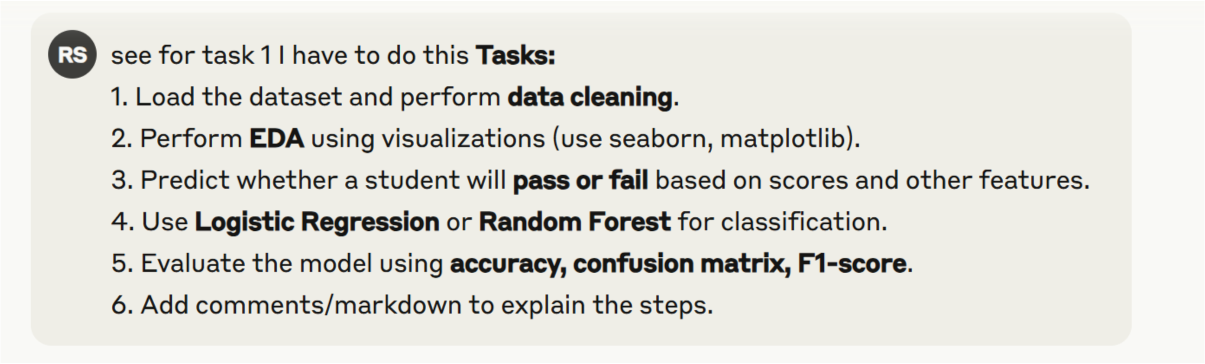
this is the first part 1 dataset

I see you've shared information about the StudentsPerformance.csv dataset for the first part of your assignment. Let's analyze this dataset and create a Python solution that covers all the requirements for Section 1.

Let's explore the dataset structure first:

Now, let's do some exploratory data analysis to understand the distributions and relationships in the data:

Now let's build and evaluate machine learning models to predict student pass/fail status:



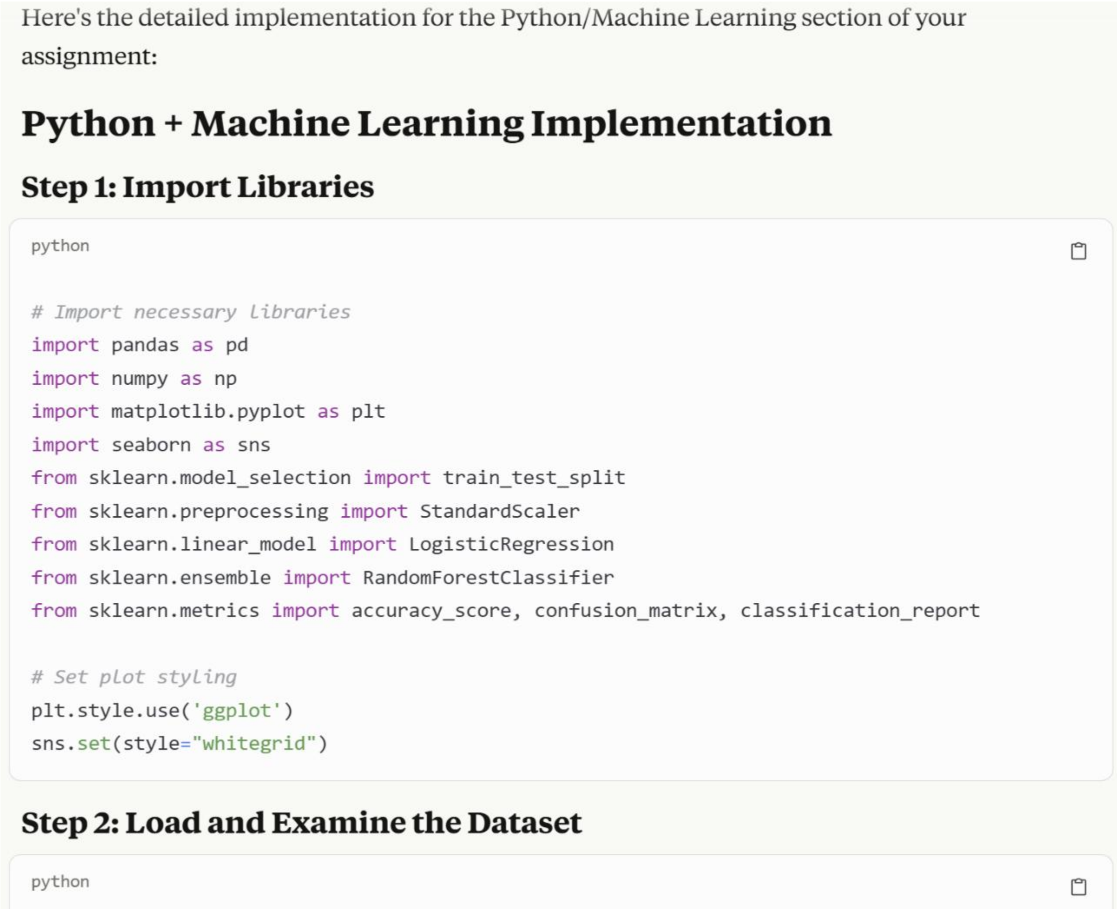
see for task 1 I have to do this **Tasks:**

1. Load the dataset and perform **data cleaning**.
2. Perform **EDA** using visualizations (use seaborn, matplotlib).
3. Predict whether a student will **pass or fail** based on scores and other features.
4. Use **Logistic Regression** or **Random Forest** for classification.
5. Evaluate the model using **accuracy, confusion matrix, F1-score**.
6. Add comments/markdown to explain the steps.

Claude’s Response

Claude provided a detailed, step-by-step implementation of the machine learning pipeline, including the following key components:

Figure 1: Step 1 - Import Libraries



Here's the detailed implementation for the Python/Machine Learning section of your assignment:

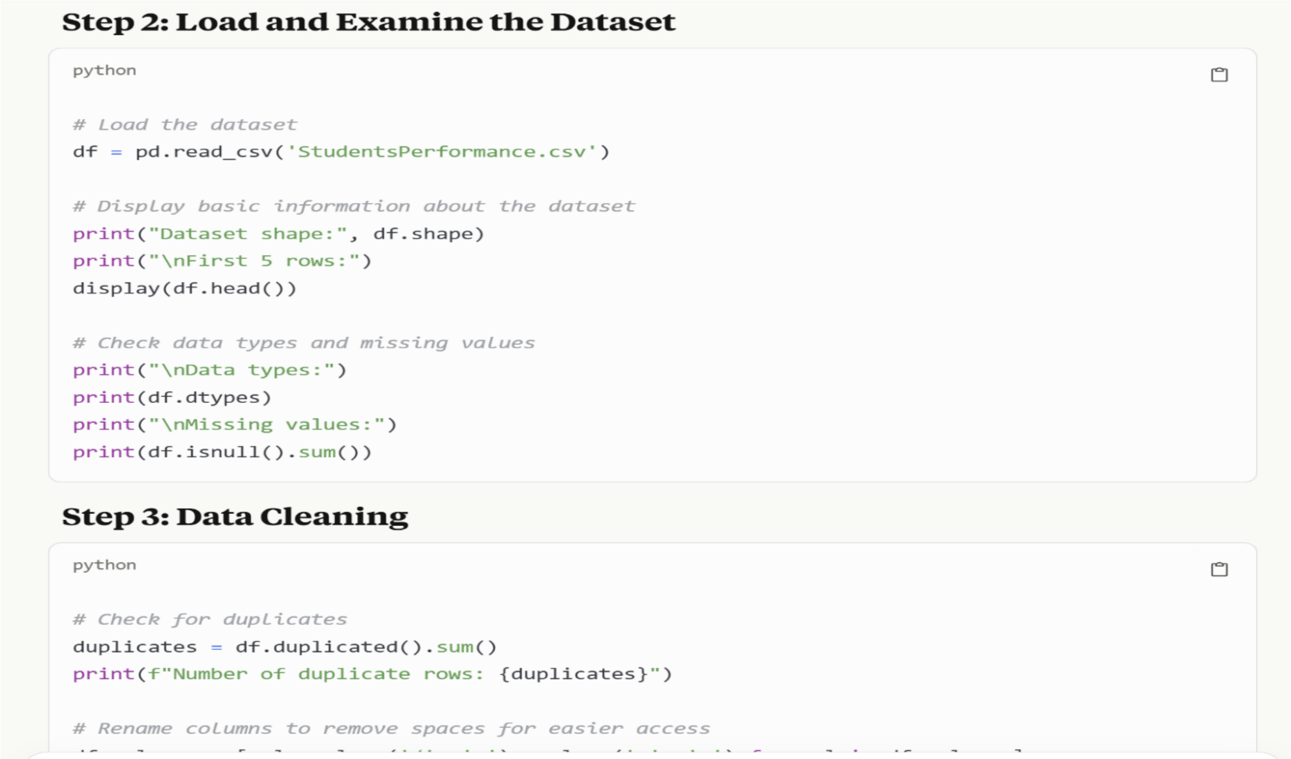
Python + Machine Learning Implementation

Step 1: Import Libraries

```python
# Import necessary 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
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Set plot styling
plt.style.use('ggplot')
sns.set(style="whitegrid")
```

Step 2: Load and Examine the Dataset

Figure 2: Step 2 - Load and Examine the Dataset



Step 2: Load and Examine the Dataset

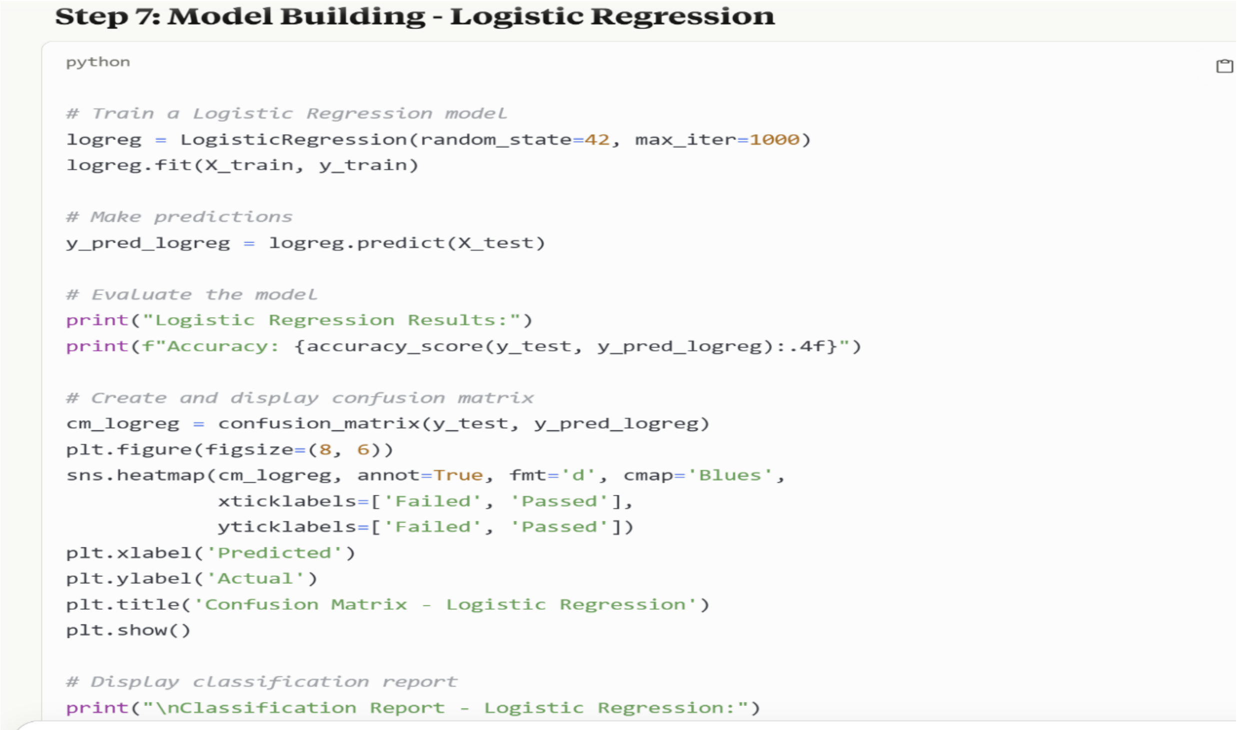
```python
# Load the dataset
df = pd.read_csv('StudentsPerformance.csv')

# Display basic information about the dataset
print("Dataset shape:", df.shape)
print("\nFirst 5 rows:")
display(df.head())

# Check data types and missing values
print("\nData types:")
print(df.dtypes)
print("\nMissing values:")
print(df.isnull().sum())
```

Step 3: Data Cleaning

```python
# Check for duplicates
duplicates = df.duplicated().sum()
print(f"Number of duplicate rows: {duplicates}")

# Rename columns to remove spaces for easier access
```

For the model building part, Claude provided comprehensive code for both Logistic Regression and Random Forest classifiers:

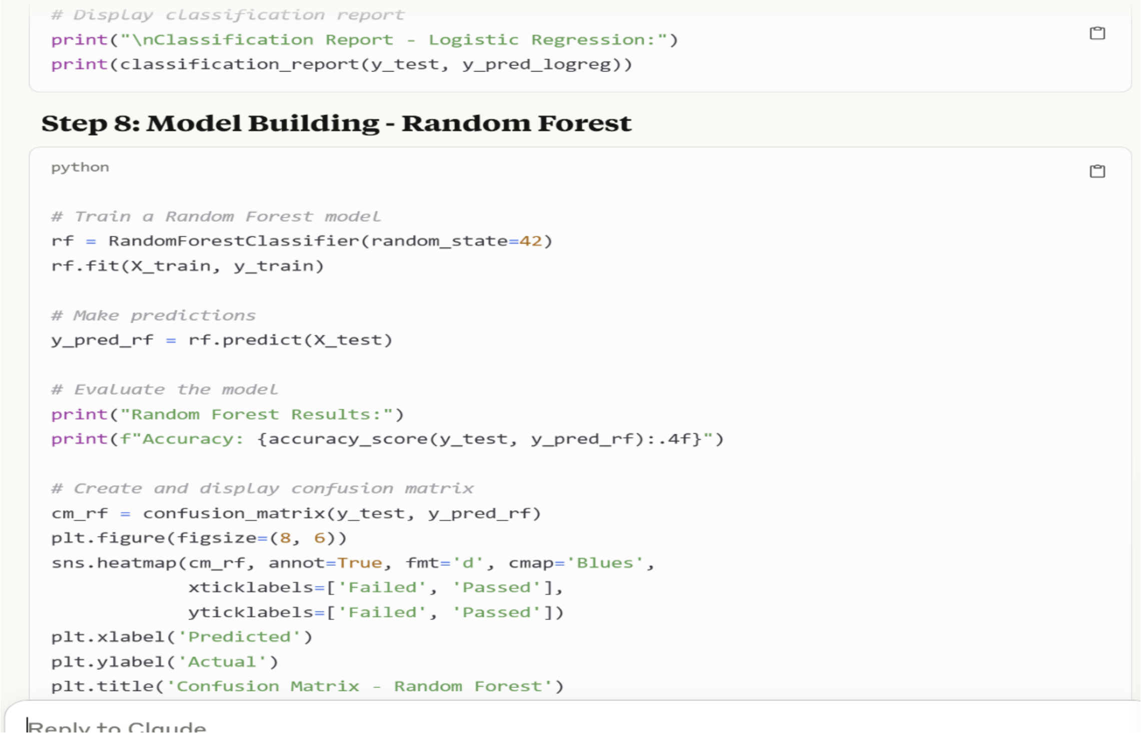
Figure 3: Step 7 - Model Building - Logistic Regression



Step 7: Model Building - Logistic Regression

```python
# Train a Logistic Regression model
logreg = LogisticRegression(random_state=42, max_iter=1000)
logreg.fit(X_train, y_train)

# Make predictions
y_pred_logreg = logreg.predict(X_test)

# Evaluate the model
print("Logistic Regression Results:")
print(f"Accuracy: {accuracy_score(y_test, y_pred_logreg):.4f}")

# Create and display confusion matrix
cm_logreg = confusion_matrix(y_test, y_pred_logreg)
plt.figure(figsize=(8, 6))
sns.heatmap(cm_logreg, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Failed', 'Passed'],
            yticklabels=['Failed', 'Passed'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - Logistic Regression')
plt.show()

# Display classification report
print("\nClassification Report - Logistic Regression:")
```

Figure 4: Steps 7-8 - Logistic Regression and Random Forest

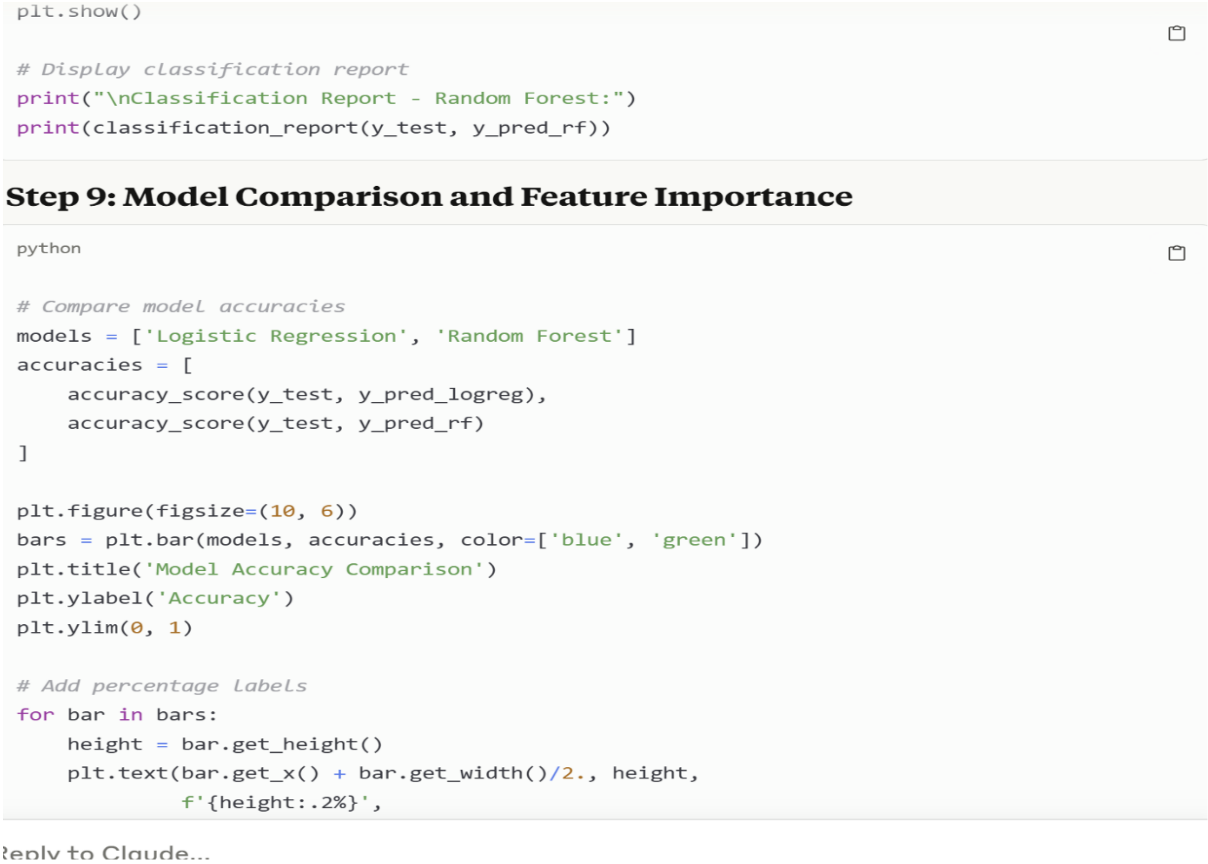


```
# Display classification report
print("\nClassification Report - Logistic Regression:")
print(classification_report(y_test, y_pred_logreg))
```

Step 8: Model Building - Random Forest

```
# Train a Random Forest model
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)

# Make predictions
y_pred_rf = rf.predict(X_test)

# Evaluate the model
print("Random Forest Results:")
print(f"Accuracy: {accuracy_score(y_test, y_pred_rf):.4f}")

# Create and display confusion matrix
cm_rf = confusion_matrix(y_test, y_pred_rf)
plt.figure(figsize=(8, 6))
sns.heatmap(cm_rf, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Failed', 'Passed'],
            yticklabels=['Failed', 'Passed'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - Random Forest')
```

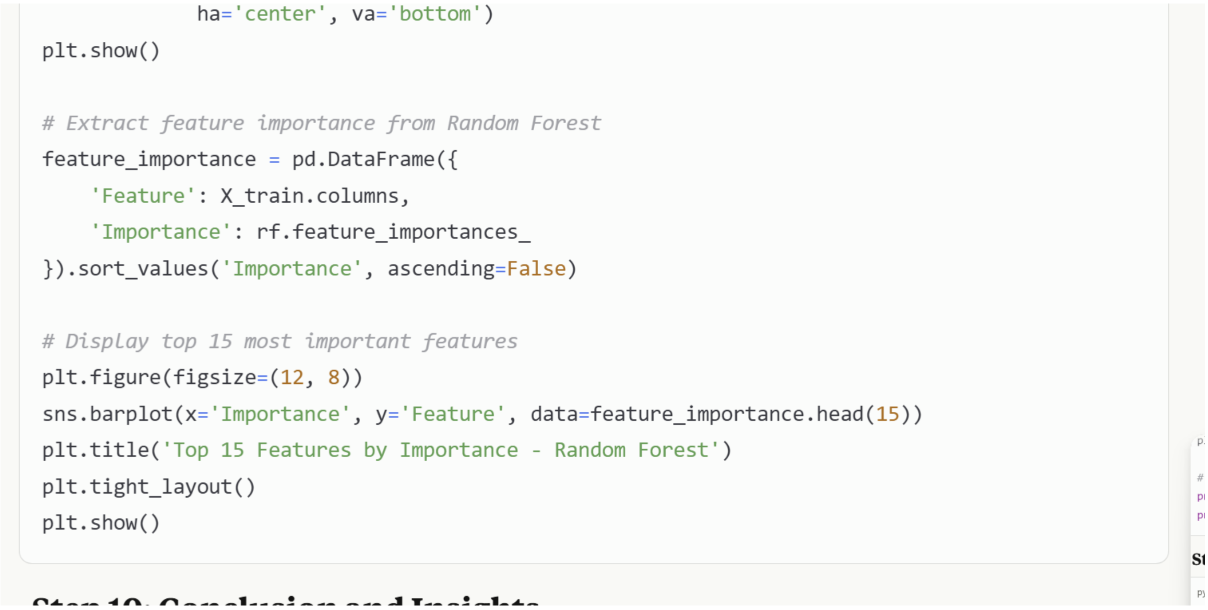
Figure 5: Steps 8-9 - Random Forest (Part 2) and Model Comparison (Part 1)



```
plt.show()

# Display classification report
print("\nClassification Report - Random Forest:")
print(classification_report(y_test, y_pred_rf))
```

Step 9: Model Comparison and Feature Importance

```
# Compare model accuracies
models = ['Logistic Regression', 'Random Forest']
accuracies = [
    accuracy_score(y_test, y_pred_logreg),
    accuracy_score(y_test, y_pred_rf)
]

plt.figure(figsize=(10, 6))
bars = plt.bar(models, accuracies, color=['blue', 'green'])
plt.title('Model Accuracy Comparison')
plt.ylabel('Accuracy')
plt.ylim(0, 1)

# Add percentage labels
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2., height,
             f'{height:.2%}',
```



```
             ha='center', va='bottom')
plt.show()

# Extract feature importance from Random Forest
feature_importance = pd.DataFrame({
    'Feature': X_train.columns,
    'Importance': rf.feature_importances_
}).sort_values('Importance', ascending=False)

# Display top 15 most important features
plt.figure(figsize=(12, 8))
sns.barplot(x='Importance', y='Feature', data=feature_importance.head(15))
plt.title('Top 15 Features by Importance - Random Forest')
plt.tight_layout()
plt.show()
```

Step 10: Conclusion and Insights

Figure 6: Step 9 - Model Comparison and Feature Importance

My Reflection

Claude provided well-structured, comprehensive code for implementing both Logistic Regression and Random Forest models, complete with proper evaluation metrics. The AI did an excellent job explaining each step through comments and organizing the code in a logical sequence. I had to modify the code slightly to adapt to my specific dataset structure and column names. I also enhanced some visualizations to make them more informative for my presentation. Overall, the AI-generated code served as an excellent starting point that saved significant development time while ensuring all required evaluation metrics were properly implemented.