Healthcare Project - Claim Approval

Build a predictive model (logistic regression model) to predict whether a claim is approved (1) or not approved (0) based on three features: "claim_amount", "age", and "provider_id". Let's break down the features and their role.

- claim_amount: The amount of the claim. This is likely a key determinant, as very high or low claim amounts might influence approval rates.
- age: The patient's age. Age might correlate with approval if younger or older patients are treated differently.
- provider_id: Encoded as dummy variables. This captures whether specific providers have different approval tendencies.

1. Load Data

```
import pandas as pd

path = "C:/Users/rvrei/Documents/Healthcare_df.csv"
healthcare_df = pd.read_csv(path)
healthcare_df.head()
```

	claim_id	patient_id	procedure_id	claim_date	claim_amount	claim_status	insurance_provider	procedure_type	pr
0	CLM0001	PAT0001	41	2024-03-29	1997.79	approved	Blue Shield	CT Scan	
1	CLM0001	PAT0001	41	2024-03-29	1997.79	approved	Blue Shield	CT Scan	
2	CLM0016	PAT0016	41	2023-09-16	1080.34	approved	Aetna	MRI	
3	CLM0016	PAT0016	41	2023-09-16	1080.34	approved	Aetna	MRI	
4	CLM0004	PAT0004	14	2023-02-03	3073.56	approved	Blue Shield	Lab Test	

5 rows × 21 columns

2. Predictive Model: Logistic Regression

```
# Modules
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report


# Prepare features and target
X = healthcare_df[['claim_amount','age','provider_id']]
y = healthcare_df['claim_status'].apply(lambda x: 1 if x=='approved' else 0)


# One-hot encoding for categorical variables: "provider_id"
X = pd.get_dummies(X, columns=['provider_id'], drop_first=True)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train logistic Regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Predict and evaluate
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.50      0.67         6
           1       0.57      1.00      0.73         4

    accuracy                           0.70        10
   macro avg       0.79      0.75      0.70        10
weighted avg       0.83      0.70      0.69        10
```

3. Results and their Significance to the input data

3.1 Logistic Regression Results

The metrics are provided for each class:

Class 0 (Not Approved)

> Precision: 1.00 When the model predicts a claim as "not approved," it is always correct. No false positives for class 0.
> Recall: 0.50 Out of all actual "not approved" claims, the model correctly identified only 50%. The recall is low, indicating false negatives (claims incorrectly predicted as approved).
> F1-Score: 0.67 This score combines precision and recall, showing moderate performance for class 0.

Class 1 (Approved)

> Precision: 0.57 When the model predicts "approved," it is correct 57% of the time. The precision is moderate, indicating false positives (claims incorrectly predicted as approved).
> Recall: 1.00 The model identified all actual "approved" claims (perfect recall).
> F1-Score: 0.73 This score is better than class 0 because recall is perfect, and precision is reasonable.

```
Accuracy: 70%
    The model correctly predicted 7 out of 10 claims.

Macro Avg:
    Precision (0.79): Average precision across both classes, treating them equally.
    Recall (0.75): Average recall across both classes, treating them equally.
    F1-Score (0.70): Average F1-score across both classes.

Weighted Avg:
```

```
Metrics weighted by the number of actual instances of each class.
These metrics are more useful when dealing with imbalanced datasets.
```

3.2 Interpretation of Results

- High Recall for Class 1 (Approved Claims): The model identifies all approved claims correctly. This is crucial in scenarios where missing approved claims has significant consequences (e.g., avoiding customer dissatisfaction).
- Low Recall for Class 0 (Not Approved Claims): The model fails to identify half of the not-approved claims. This might lead to approving claims that should not be approved.
- Feature Impact
  - claim_amount: Likely a strong predictor. Large claim amounts might lead to more scrutiny, influencing approval likelihood.
  - age: Its impact depends on the correlation in the dataset. If certain age groups are more likely to have approved claims, the model will reflect this.
  - provider_id: Dummy variables allow the model to detect patterns specific to providers.
- Business Insights: If avoiding false negatives for approved claims (class 1) is crucial, the model performs well with 100% recall for this class. However, the low recall for not-approved claims (class 0) indicates a risk of approving some inappropriate claims. Use the model to prioritize claims for manual review or set policies to handle cases flagged as uncertain.

3.3 Steps to Improve the Model

- Handle Class Imbalance: If "not approved" claims dominate, the model might struggle with recall for this class. Techniques: oversample the minority class, undersample the majority class, or use class weights.
- Feature Engineering: Explore non-linear relationships between features and the target variable. Add interaction terms, such as

claim_amount × provider_id.

- Threshold Tuning: The model's default threshold is 0.5. Adjusting this threshold can improve recall or precision, depending on the priority.
- Evaluate Additional Models: decision trees, random forests, or gradient boosting for potentially better performance.

4. Model Visualization: Logistic Regression

Visualizations to evaluate the Claim Approval Model using logistic regression results.

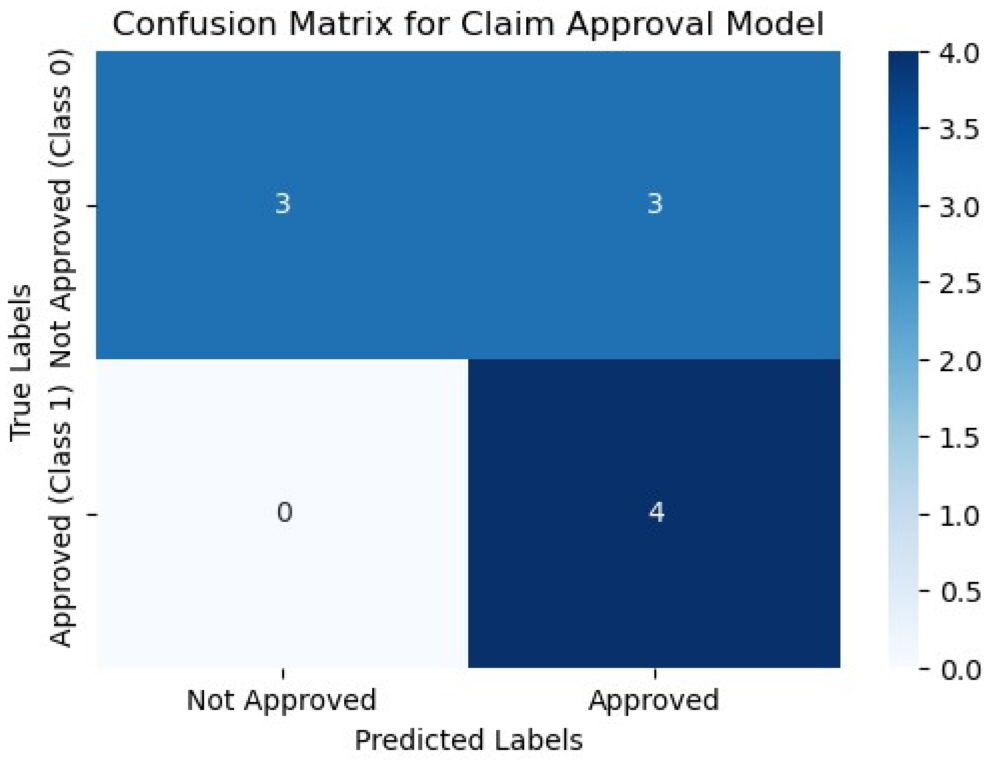
```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, roc_curve, roc_auc_score, precision_recall_curve
from sklearn.linear_model import LogisticRegression
```

4.1 Confusion Matrix

This shows how well the model is performing, i.e., how many claims were correctly and incorrectly classified. The confusion matrix compares the actual values (y_test) against the predicted values (y_pred) to show how well the model is performing.

```
from sklearn.metrics import confusion_matrix, accuracy_score

# Generate confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Create a heatmap for visualization
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',  xticklabels=['Not Approved', 'Approved'],\
            yticklabels=['Not Approved (Class 0)', 'Approved (Class 1)'])
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix for Claim Approval Model')

# Calculate and print accuracy
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy)

plt.show()
```

```
Accuracy: 0.7
```



This heatmap visualizes the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). It helps in evaluating the model's accuracy.

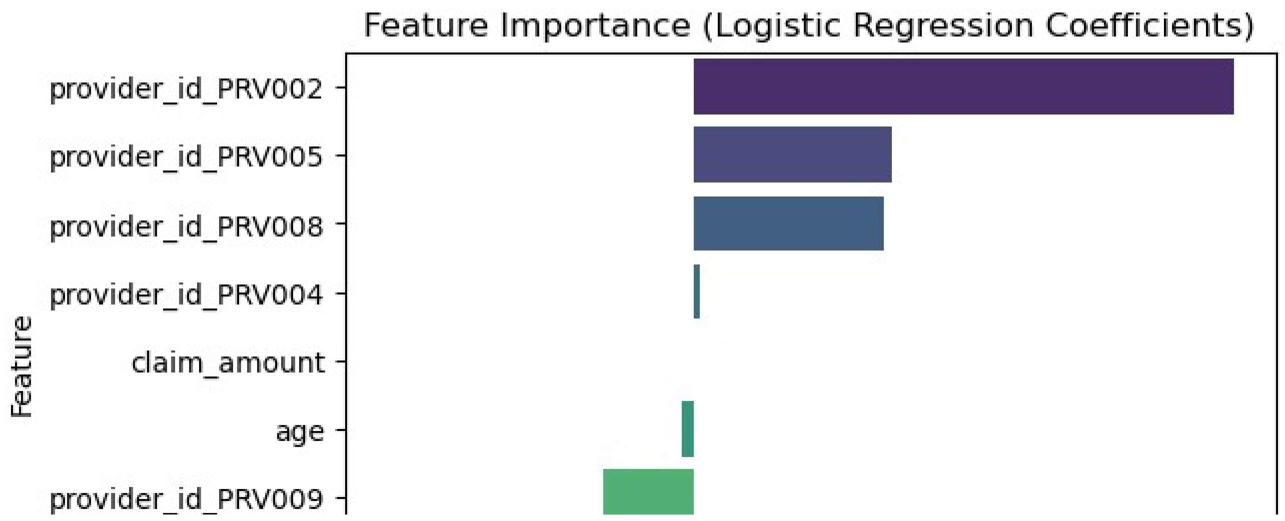
- Diagonal values: Correct classifications. A high number on the diagonal (True Positives and True Negatives) means the model is classifying claims well.

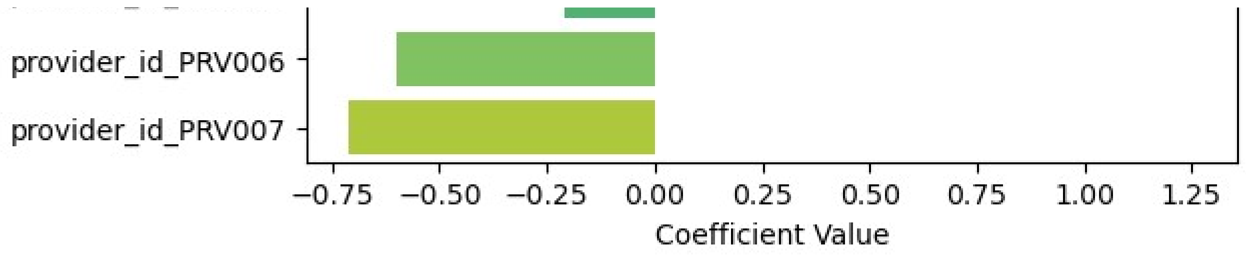
- Off-diagonal values: Incorrect classifications. A high number off-diagonal (False Positives or False Negatives) means the model is making more mistakes.
- Labels: "No Approved (Class 0)" and "Approved (Class 1)" are the actual classes.

4.2 Feature Importance

Feature Importance (Coefficients): This shows the relative importance of each feature used by the logistic regression model.

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

plt.figure(figsize=(6, 4))
sns.barplot(x='Coefficient', y='Feature', data=coefficients, palette='viridis')
plt.title('Feature Importance (Logistic Regression Coefficients)')
plt.xlabel('Coefficient Value')
plt.ylabel('Feature')
plt.show()
```



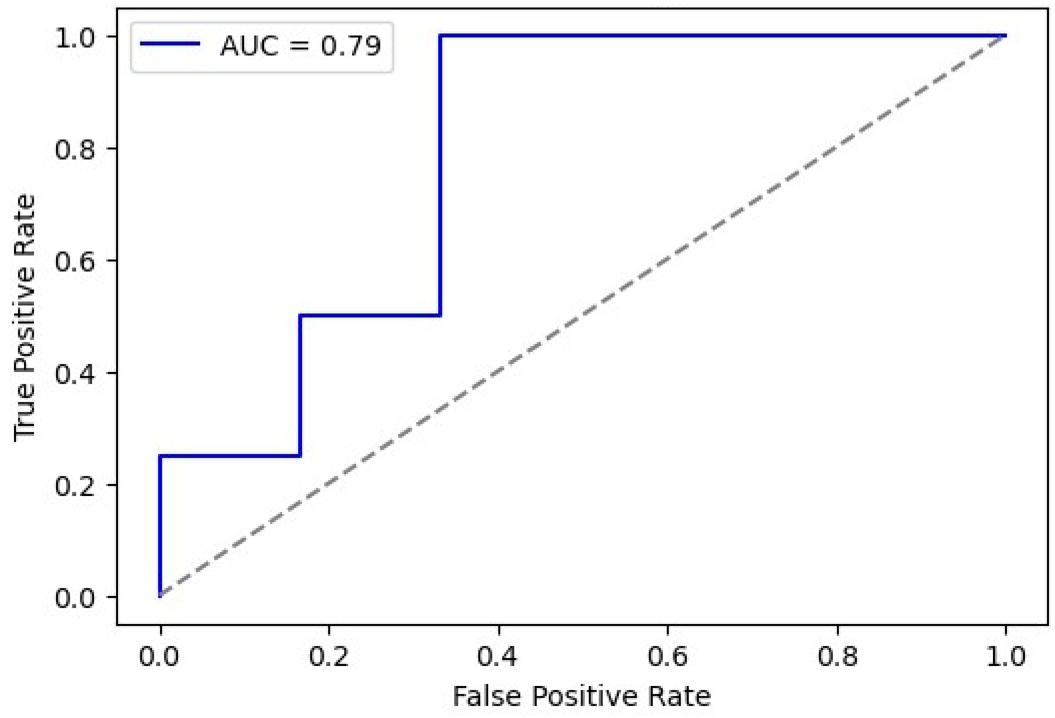


This bar plot shows the coefficients for each feature in the logistic regression model. Features with large coefficients (positive or negative) have a bigger influence on whether a claim gets approved or not. The sign of the coefficient tells whether the feature increases or decreases the probability of a "Yes" (claim approval).

4.3 ROC Curve

This evaluates the trade-off between the True Positive Rate (Recall) and the False Positive Rate.

```
# ROC Curve
y_prob = model.predict_proba(X_test)[:, 1]  # probability of 'Yes' (1)

fpr, tpr, thresholds = roc_curve(y_test, y_prob)
auc = roc_auc_score(y_test, y_prob)

plt.figure(figsize=(6, 4))
plt.plot(fpr, tpr, color='blue', label=f'AUC = {auc:.2f}')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')  # Diagonal line for random classifier
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for Claim Approval Model')
plt.legend()
plt.show()
```

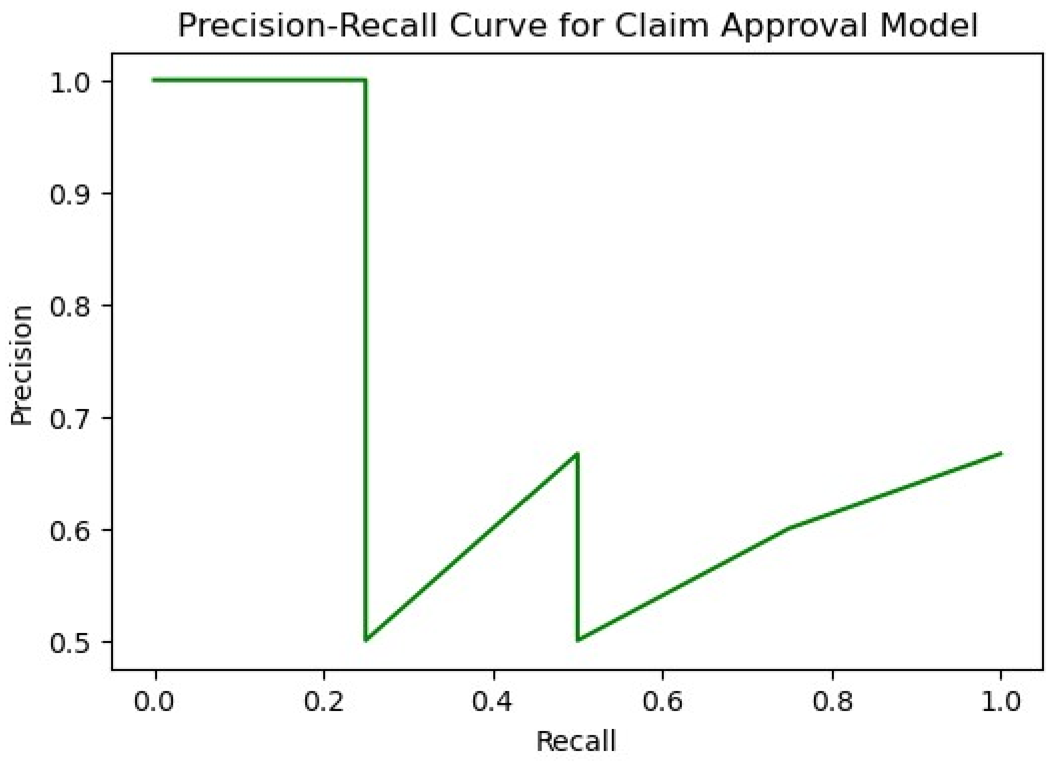


- The ROC curve plots the True Positive Rate (recall) against the False Positive Rate. The closer the curve is to the top-left corner, the better the model.
- AUC (Area Under Curve): Measures the overall ability of the model to distinguish between classes. A value closer to 1 indicates a good model. A model with a high AUC is better at distinguishing between the classes (perfect model would have an AUC of 1.0).

4.4 Precision-Recall Curve

This gives a better understanding of the model performance for imbalanced classes (if your dataset has more "No" than "Yes" claims).

```
# Precision-Recall Curve
precision, recall, _ = precision_recall_curve(y_test, y_prob)

plt.figure(figsize=(6, 4))
plt.plot(recall, precision, color='green')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve for Claim Approval Model')
plt.show()
```



This curve illustrates the trade-off between precision and recall at various thresholds. In the context of an imbalanced dataset—where one class (e.g., "No" claims) dominates over the other (e.g., fewer "Yes" claims)—the curve highlights the model's ability to identify the minority class effectively. It provides valuable insights into the model's performance specifically for the minority class (e.g., "Yes").

(e.g., "Yes").

If your primary objective is to minimize false positives, you should focus on achieving higher precision.

5. Prediction for a new patient: Logistic Regression Model

Predict if a claim would be approved or not approved for a new patient with a claim amount of $2,500, 65 years old, and with the provider PVR003.

To use the logistic regression model to predict for a new patient, we need to ensure that the input data for the prediction has the same structure as the training data. Specifically:

- Handle Dummy Variables: Since provider_id was one-hot encoded during preprocessing, we must represent it in the same way for the new data. This ensures that the new data has the same columns as the training data.
- Format the Input: The input data must be in the form of a DataFrame or array with the same columns as the original X used during training.

```
import pandas as pd

# New patient data
new_patient = pd.DataFrame({
    'claim_amount': [2500],
    'age': [65],
    'provider_id': ['PVR003']
})

# One-hot encode the new patient data
new_patient_encoded = pd.get_dummies(new_patient, columns=['provider_id'])

# Align the new patient columns with the training data columns
new_patient_encoded = new_patient_encoded.reindex(columns=X.columns, fill_value=0)

# Predict using the trained model
```

```
prediction = model.predict(new_patient_encoded)

# Output the result
if prediction[0] == 1:
    print("The claim is predicted to be approved.")
else:
    print("The claim is predicted to be denied.")
```

```
The claim is predicted to be approved.
```