Report

Healthcare Predictive Analytics

Project Name:

Healthcare Predictive Analytics

Dataset Source:

Gastric Cancer (GC) Dataset - Kaggle
(https://www.kaggle.com/datasets/datasetengineer/gastric-cancer-gc-dataset)

GitHub Repository:

Depi Project - GitHub (https://github.com/7ayder-99/depi-project)

Project Overview:

The 'Healthcare Predictive Analytics' project focuses on using machine learning techniques to predict health outcomes based on patient data. Specifically, the project utilizes the Gastric Cancer
(GC) dataset available on Kaggle. The primary objective is to develop predictive models that can
assist healthcare professionals in early diagnosis, treatment planning, and improving patient outcomes.

Dataset Description:

The Gastric Cancer dataset contains a variety of clinical features related to patients diagnosed with
gastric cancer. These features include demographic information, clinical test results, and pathological findings. The dataset provides a valuable resource for building predictive models aimed
at understanding risk factors and predicting patient prognosis.

Project Progress

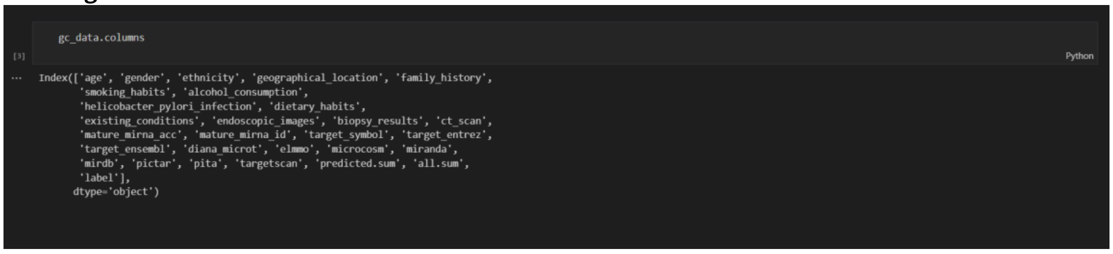
Milestone 1: Data Collection, Exploration, and Preprocessing

- ***Data Collection:***

Obtain healthcare dataset : Gastric Cancer (GC) Dataset - Kaggle (https://www.kaggle.com/datasets/datasetengineer/gastric-cancer-gc-dataset)

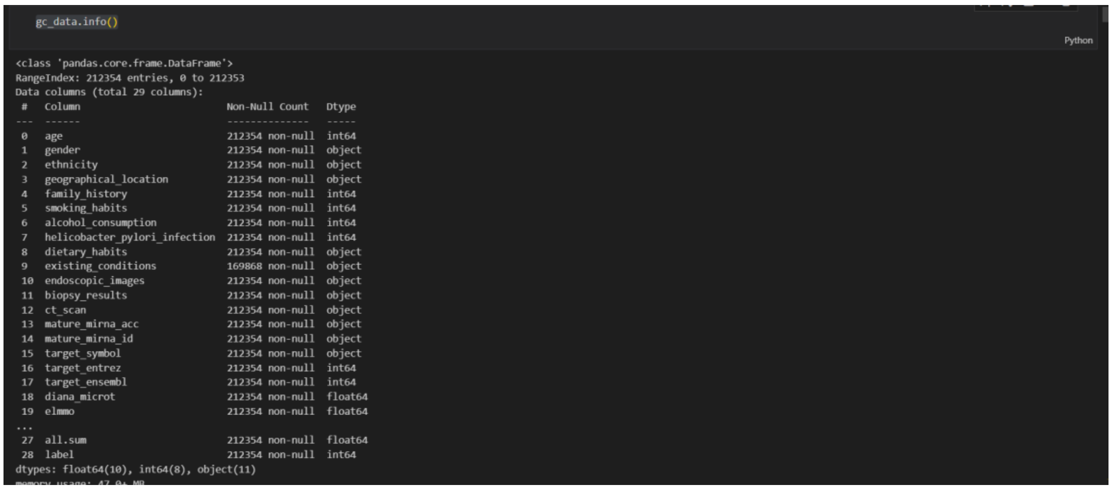
- ***Data Exploration:***

Reading colums :



```
gc_data.columns
[3]                                                                                         Python

... Index(['age', 'gender', 'ethnicity', 'geographical_location', 'family_history',
       'smoking_habits', 'alcohol_consumption',
       'helicobacter_pylori_infection', 'dietary_habits',
       'existing_conditions', 'endoscopic_images', 'biopsy_results', 'ct_scan',
       'mature_mirna_acc', 'mature_mirna_id', 'target_symbol', 'target_entrez',
       'target_ensembl', 'diana_microt', 'elmmo', 'microcosm', 'miranda',
       'mirdb', 'pictar', 'pita', 'targetscan', 'predicted.sum', 'all.sum',
       'label'],
      dtype='object')
```

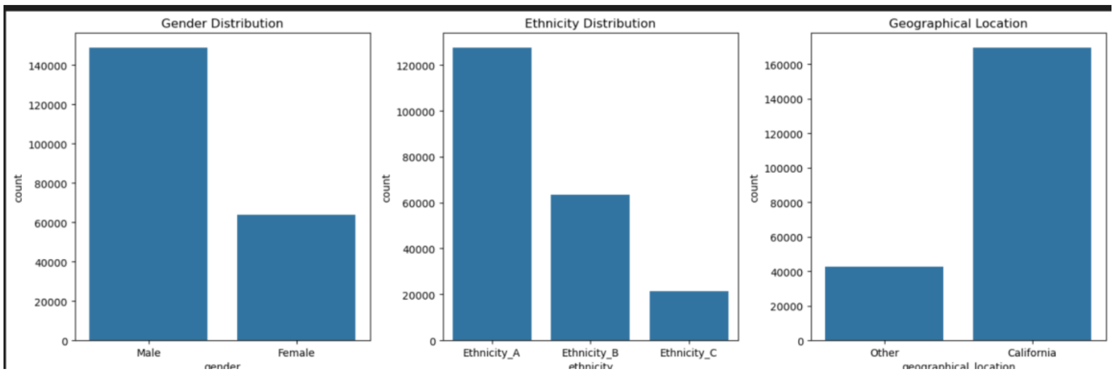
Detailed info of Data :

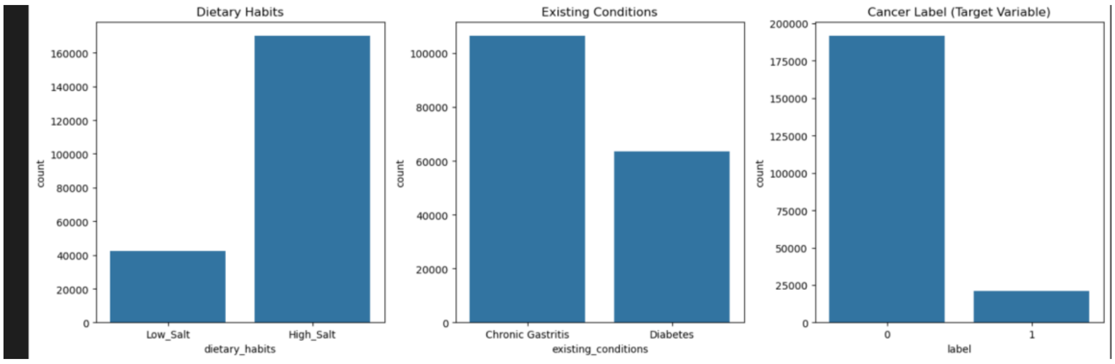


```
gc_data.info()
                                                                                            Python
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 212354 entries, 0 to 212353
Data columns (total 29 columns):
 #   Column                         Non-Null Count   Dtype
---  ------                         --------------   -----
 0   age                            212354 non-null  int64
 1   gender                         212354 non-null  object
 2   ethnicity                      212354 non-null  object
 3   geographical_location          212354 non-null  object
 4   family_history                 212354 non-null  int64
 5   smoking_habits                 212354 non-null  int64
 6   alcohol_consumption            212354 non-null  int64
 7   helicobacter_pylori_infection  212354 non-null  int64
 8   dietary_habits                 212354 non-null  object
 9   existing_conditions            169868 non-null  object
 10  endoscopic_images              212354 non-null  object
 11  biopsy_results                 212354 non-null  object
 12  ct_scan                        212354 non-null  object
 13  mature_mirna_acc               212354 non-null  object
 14  mature_mirna_id                212354 non-null  object
 15  target_symbol                  212354 non-null  object
 16  target_entrez                  212354 non-null  int64
 17  target_ensembl                 212354 non-null  int64
 18  diana_microt                   212354 non-null  float64
 19  elmmo                          212354 non-null  float64
...
 27  all.sum                        212354 non-null  float64
 28  label                          212354 non-null  int64
dtypes: float64(10), int64(8), object(11)
```

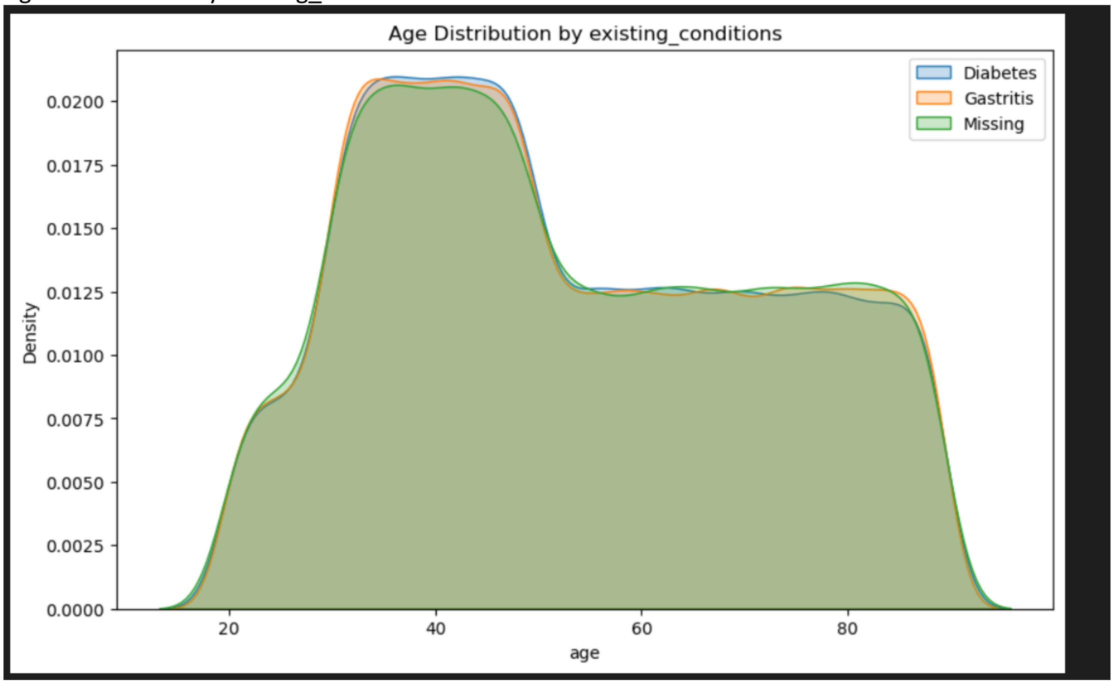
EDA :

Columns Distributions :-

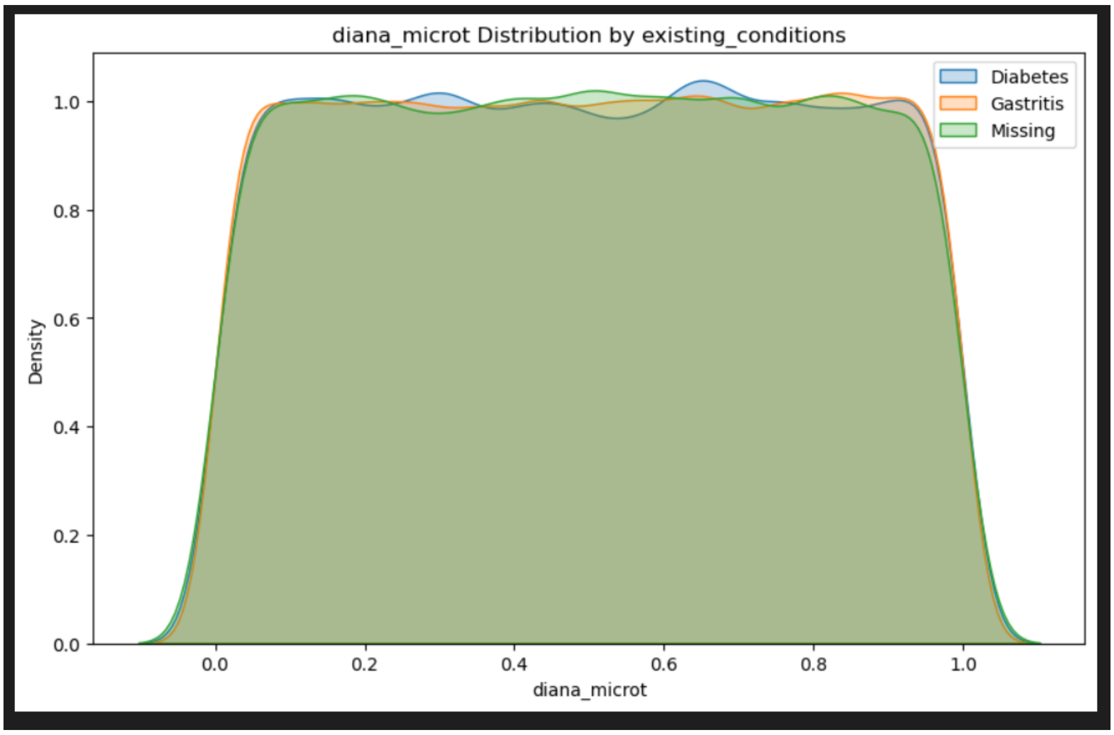




Age distributions by existing_conditions :

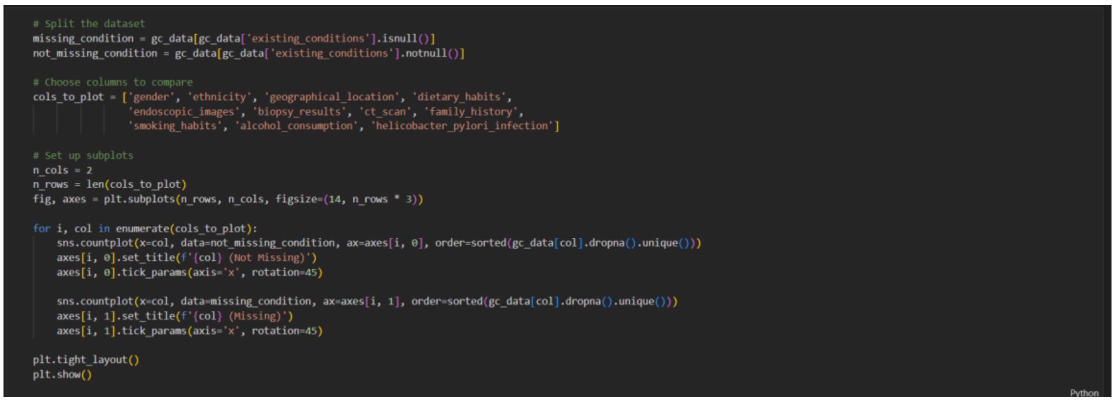


Diana_microt distribution :-

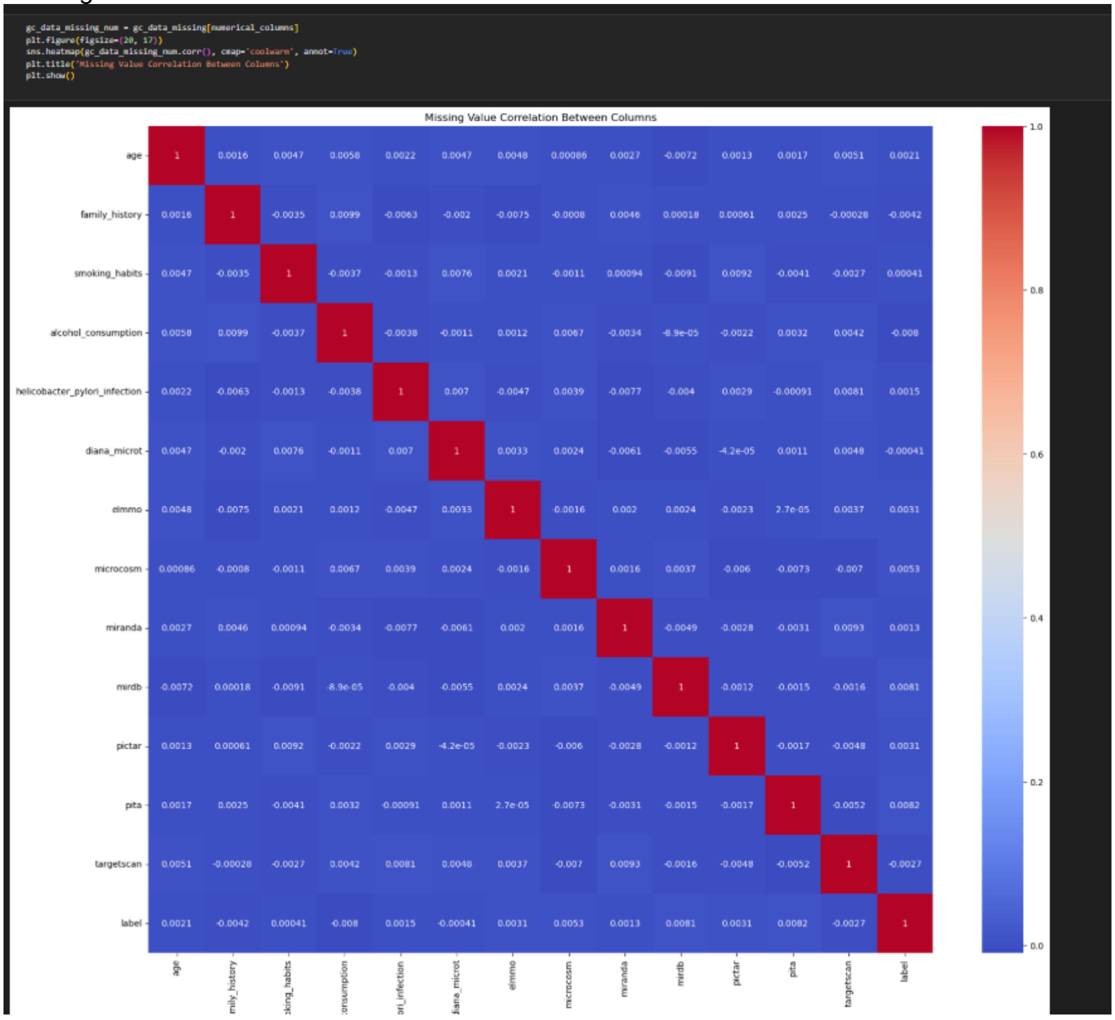


- Data Preprocessing

Spliting the dataset :-

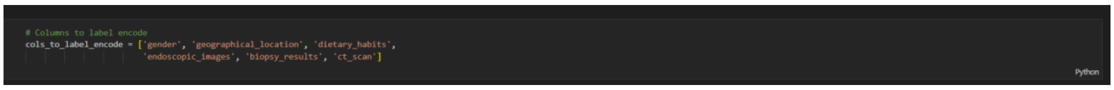

```python
# Split the dataset
missing_condition = gc_data[gc_data['existing_conditions'].isnull()]
not_missing_condition = gc_data[gc_data['existing_conditions'].notnull()]

# Choose columns to compare
cols_to_plot = ['gender', 'ethnicity', 'geographical_location', 'dietary_habits',
                'endoscopic_images', 'biopsy_results', 'ct_scan', 'family_history',
                'smoking_habits', 'alcohol_consumption', 'helicobacter_pylori_infection']

# Set up subplots
n_cols = 2
n_rows = len(cols_to_plot)
fig, axes = plt.subplots(n_rows, n_cols, figsize=(14, n_rows * 3))

for i, col in enumerate(cols_to_plot):
    sns.countplot(x=col, data=not_missing_condition, ax=axes[i, 0], order=sorted(gc_data[col].dropna().unique()))
    axes[i, 0].set_title(f'{col} (Not Missing)')
    axes[i, 0].tick_params(axis='x', rotation=45)

    sns.countplot(x=col, data=missing_condition, ax=axes[i, 1], order=sorted(gc_data[col].dropna().unique()))
    axes[i, 1].set_title(f'{col} (Missing)')
    axes[i, 1].tick_params(axis='x', rotation=45)

plt.tight_layout()
plt.show()
```

Missing values correlation between columns :

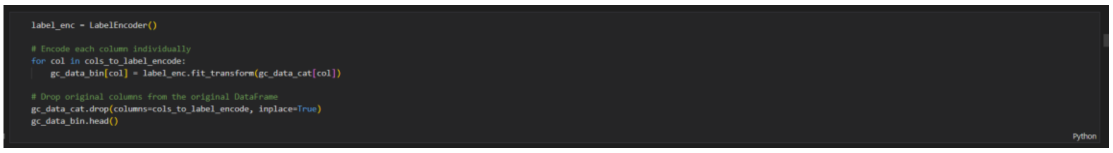


```
gc_data_missing_num = gc_data_missing[numerical_columns]
plt.figure(figsize=(20, 17))
sns.heatmap(gc_data_missing_num.corr(), cmap='coolwarm', annot=True)
plt.title('Missing Value Correlation Between Columns')
plt.show()
```

Columns to label encode :



```
# Columns to label encode
cols_to_label_encode = ['gender', 'geographical_location', 'dietary_habits',
                        'endoscopic_images', 'biopsy_results', 'ct_scan']
```

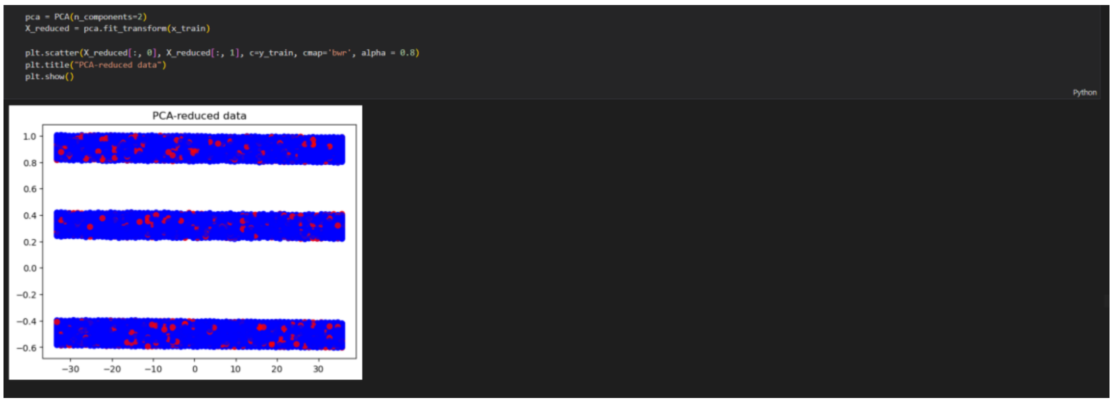


```
label_enc = LabelEncoder()

# Encode each column individually
for col in cols_to_label_encode:
    gc_data_bin[col] = label_enc.fit_transform(gc_data_cat[col])

# Drop original columns from the original DataFrame
gc_data_cat.drop(columns=cols_to_label_encode, inplace=True)
gc_data_bin.head()
```

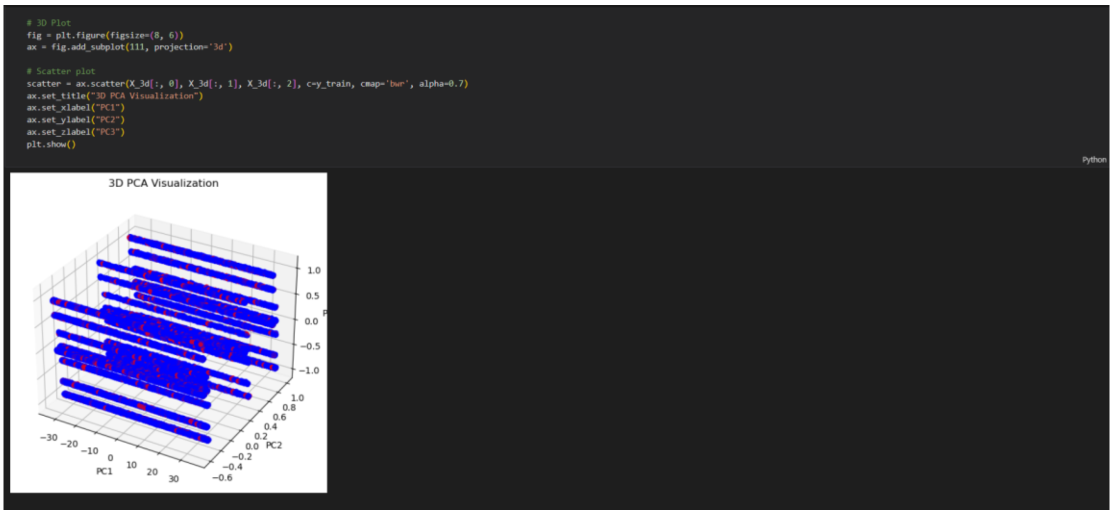
Results :

- Data is imbalanced
- Data has missing values
- Data is too complex for analysis

Data Visualization :

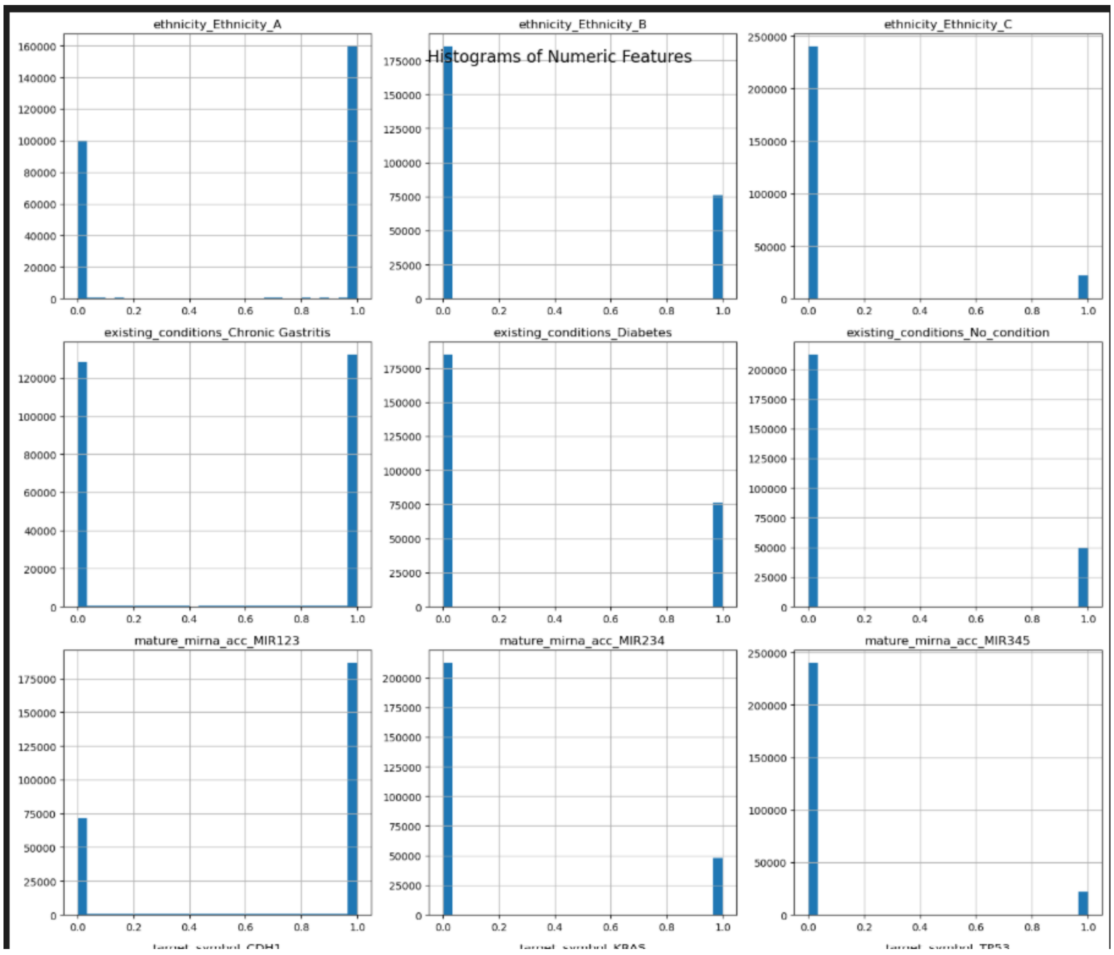


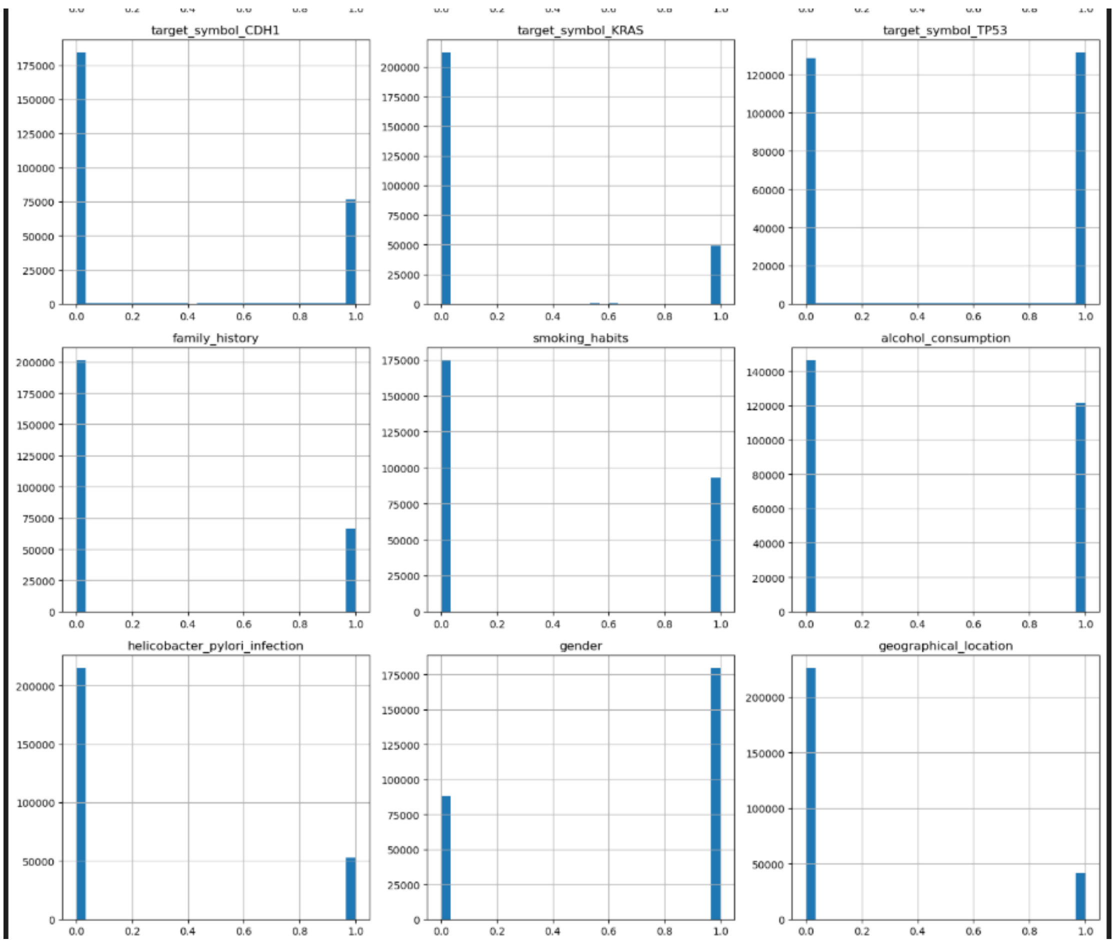
```
pca = PCA(n_components=2)
X_reduced = pca.fit_transform(x_train)

plt.scatter(X_reduced[:, 0], X_reduced[:, 1], c=y_train, cmap='bwr', alpha = 0.8)
plt.title("PCA-reduced data")
plt.show()
```

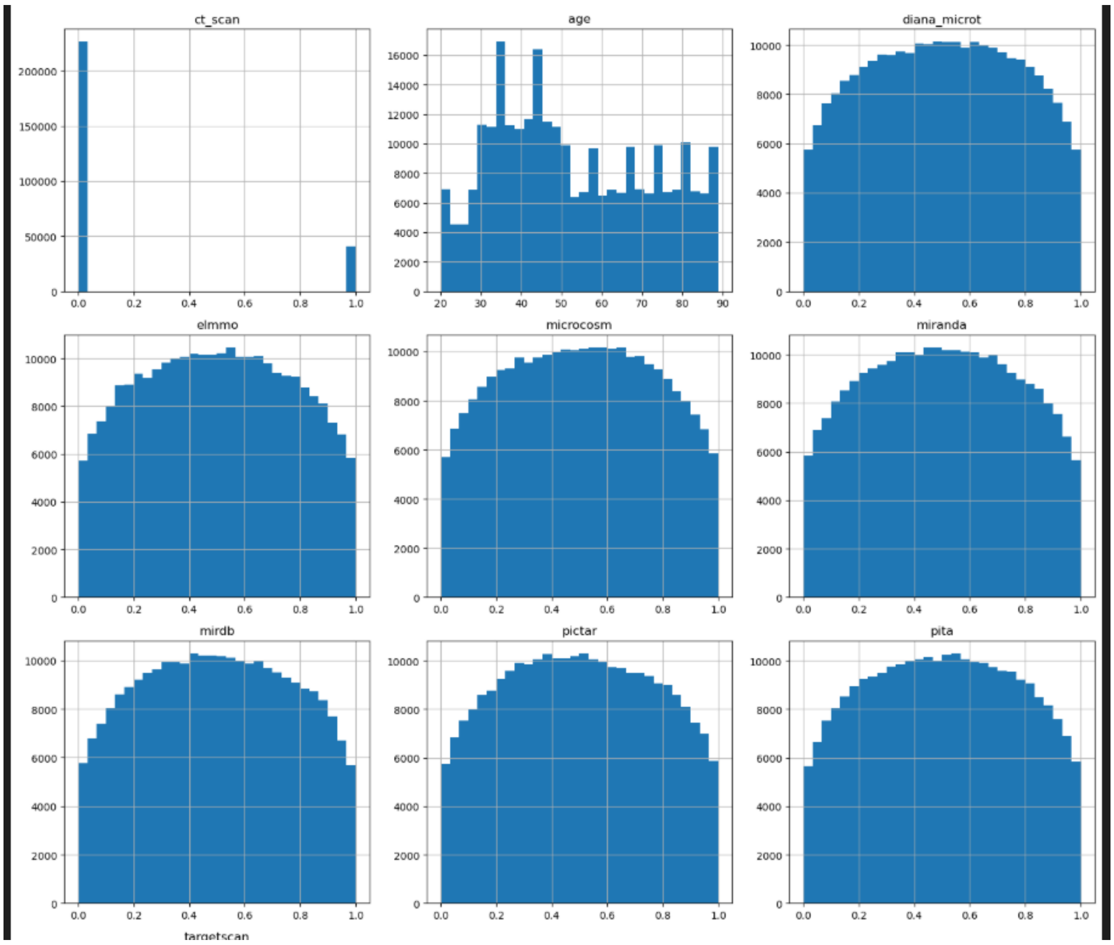


```
# 3D Plot
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')

# Scatter plot
scatter = ax.scatter(X_3d[:, 0], X_3d[:, 1], X_3d[:, 2], c=y_train, cmap='bwr', alpha=0.7)
ax.set_title("3D PCA Visualization")
ax.set_xlabel("PC1")
ax.set_ylabel("PC2")
ax.set_zlabel("PC3")
plt.show()
```

Columns after smote :

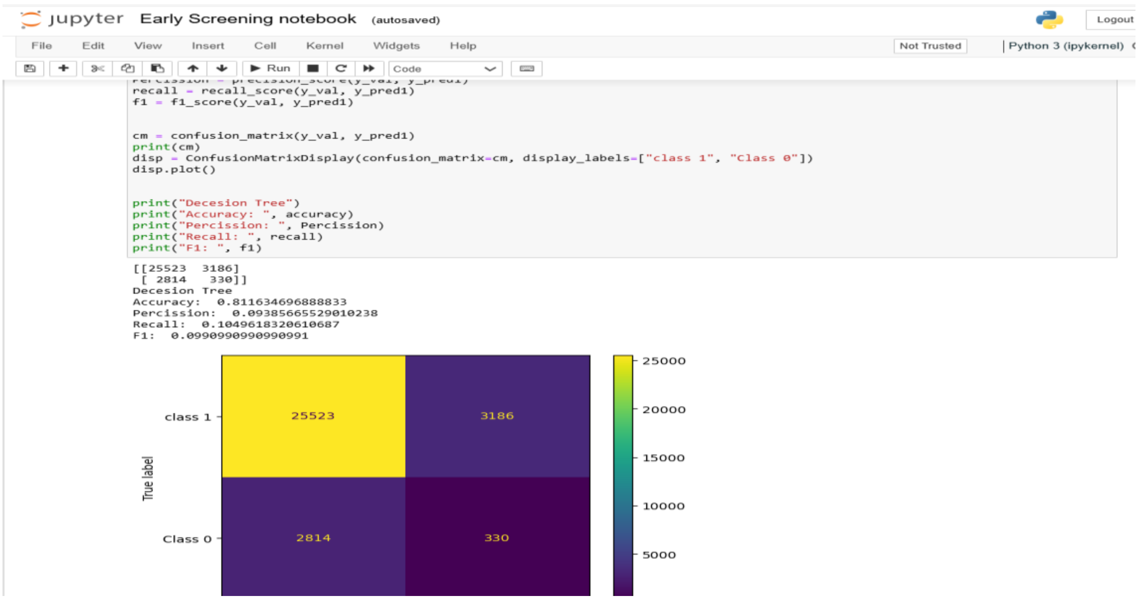



target_symbol_CDH1
target_symbol_KRAS
target_symbol_TP53
family_history
smoking_habits
alcohol_consumption
helicobacter_pylori_infection
gender
geographical_location


ct_scan
age
diana_microt
elmmo
microcosm
miranda
mirdb
pictar
pita
targetscan

Modeling :
Model 1 : Early Screening

1.Model 1 .1



```
Percission = precision_score(y_val, y_pred1)
recall = recall_score(y_val, y_pred1)
f1 = f1_score(y_val, y_pred1)


cm = confusion_matrix(y_val, y_pred1)
print(cm)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["class 1", "Class 0"])
disp.plot()


print("Decesion Tree")
print("Accuracy: ", accuracy)
print("Percission: ", Percission)
print("Recall: ", recall)
print("F1: ", f1)
```

```
[[25523  3186]
 [ 2814   330]]
Decesion Tree
Accuracy:  0.8116346968888833
Percission:  0.09385665529010238
Recall:  0.1049618320610687
F1:  0.0990990990990991
```

The Algorithm:

- **DT1** uses a **Decision Tree Classifier**, which is a supervised machine learning algorithm that classifies data based on a series of decision rules structured like a tree.
  The parameter (class_weight='balanced') was used to address class imbalance (i.e., more non-cancer cases than cancer cases), helping the model give more attention to the minority class (cancer cases).

Confusion Matrix:

- **True Negative (25523):** Healthy patients correctly predicted as healthy.
- **False Positive (3186):** Healthy patients incorrectly predicted as having cancer.
- **False Negative (2814):** Cancer patients incorrectly predicted as healthy (**high risk**).
- **True Positive (330):** Cancer patients correctly identified.

  .

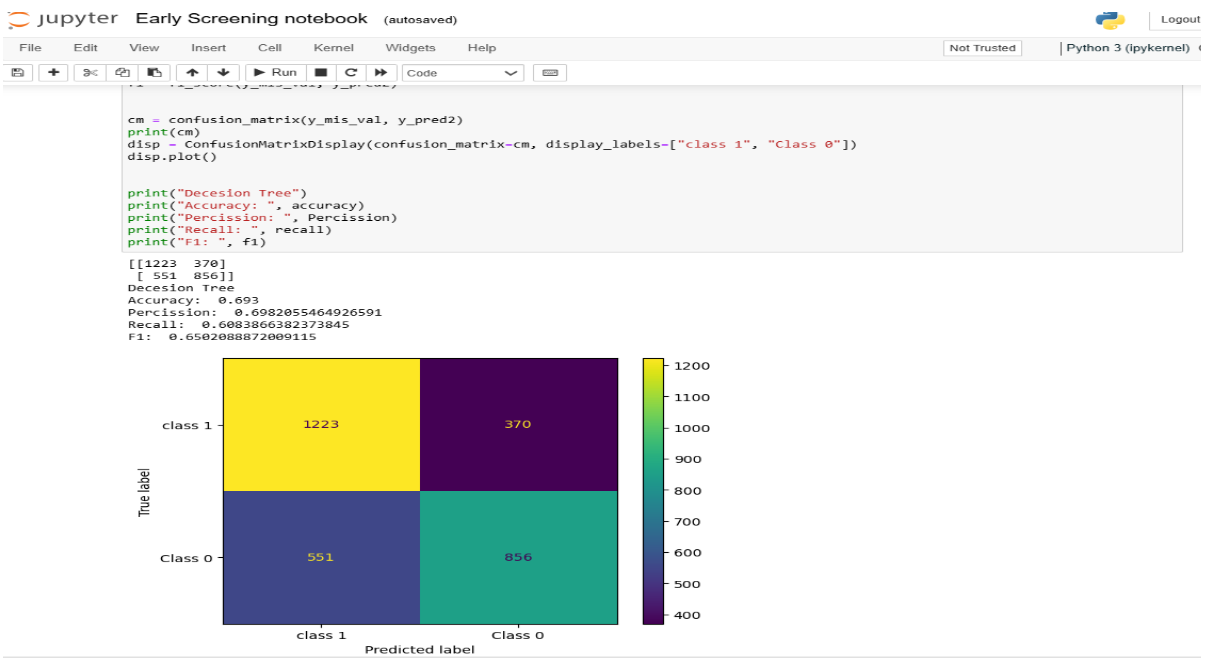
Performance Metrics:

- **Accuracy:** 81.16% — overall correct predictions, but misleading due to class imbalance.
- **Precision:** 9.39% — only 9% of those predicted as having cancer were actually correct.
- **Recall:** 10.5% — the model detected just 10% of real cancer cases, which is **very low**.
- **F1 Score:** 9.91% — the harmonic mean of precision and recall, confirming poor performance on cancer detection.

Conclusion:

- The **Decision Tree model fails to reliably detect gastric cancer**.
- **High accuracy** is misleading due to **class imbalance** — the model mostly predicts "no cancer" correctly, but fails to catch cancer cases.
- **Low recall** means the model **misses a lot of actual cancer patients**, which is dangerous for early screening.

2- Model1.2



```
cm = confusion_matrix(y_mis_val, y_pred2)
print(cm)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["class 1", "Class 0"])
disp.plot()


print("Decesion Tree")
print("Accuracy: ", accuracy)
print("Percission: ", Percission)
print("Recall: ", recall)
print("F1: ", f1)
```

```
[[1223  370]
 [ 551  856]]
Decesion Tree
Accuracy:  0.693
Percission:  0.6982055464926591
Recall:  0.6083866382373845
F1:  0.6502088872009115
```

Confusion Matrix:

- **True Negative (1223):** Healthy individuals correctly predicted as healthy.
- **False Positive (370):** Healthy individuals wrongly predicted as having cancer.
- **False Negative (551):** Cancer patients wrongly predicted as healthy.
- **True Positive (856):** Cancer patients correctly identified.

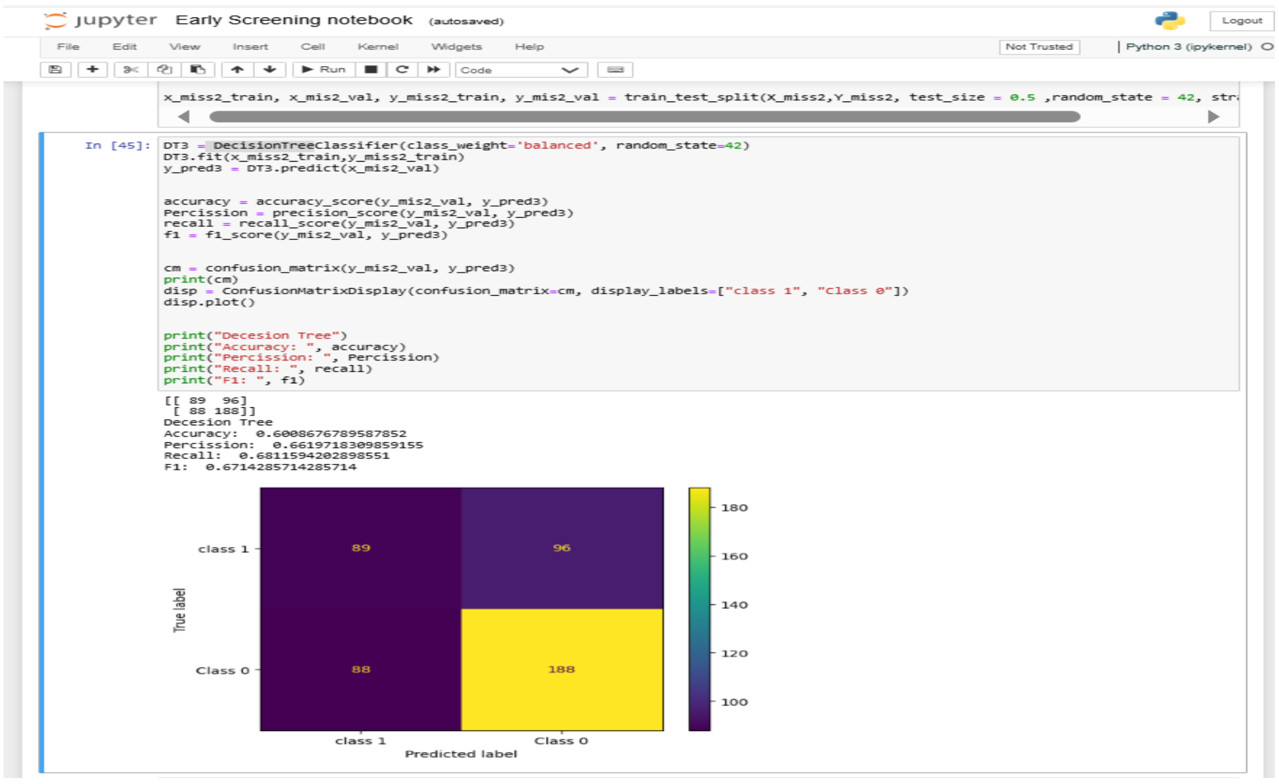
Performance Metrics:

- **Accuracy:** 69.3% — about 7 out of 10 predictions were correct.
- **Precision:** 69.82% — nearly 70% of the patients predicted as having cancer actually did.
- **Recall:** 60.84% — the model identified around 61% of all actual cancer cases.
- **F1 Score:** 65.02% — a good balance between precision and recall.

Conclusion:

- **DT2 shows a much better ability to detect cancer cases** compared to DT1.
- The **recall of 60.84%** indicates that the model catches a majority of actual cancer cases, making it more reliable for early screening.
- **Precision is also strong (69.8%)**, meaning the model avoids many false alarms.
- Overall, **DT2 provides a good balance between catching cancer cases and minimizing false positives**, which is critical in a healthcare screening

3- Model1.3



```
x_miss2_train, x_mis2_val, y_miss2_train, y_mis2_val = train_test_split(X_miss2,Y_miss2, test_size = 0.5 ,random_state = 42, str
```

```
In [45]: DT3 = DecisionTreeClassifier(class_weight='balanced', random_state=42)
DT3.fit(x_miss2_train,y_miss2_train)
y_pred3 = DT3.predict(x_mis2_val)

accuracy = accuracy_score(y_mis2_val, y_pred3)
Percission = precision_score(y_mis2_val, y_pred3)
recall = recall_score(y_mis2_val, y_pred3)
f1 = f1_score(y_mis2_val, y_pred3)

cm = confusion_matrix(y_mis2_val, y_pred3)
print(cm)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["class 1", "Class 0"])
disp.plot()

print("Decesion Tree")
print("Accuracy: ", accuracy)
print("Percission: ", Percission)
print("Recall: ", recall)
print("F1: ", f1)
```

```
[[ 89  96]
 [ 88 188]]
Decesion Tree
Accuracy:  0.6008676789587852
Percission:  0.6619718309859155
Recall:  0.6811594202898551
F1:  0.6714285714285714
```

Confusion Matrix:

- **True Negative (89):** Healthy individuals correctly classified.
- **False Positive (96):** Healthy individuals incorrectly predicted as having cancer.
- **False Negative (88):** Cancer cases incorrectly predicted as healthy.
- **True Positive (188):** Cancer cases correctly detected.

Performance Metrics:

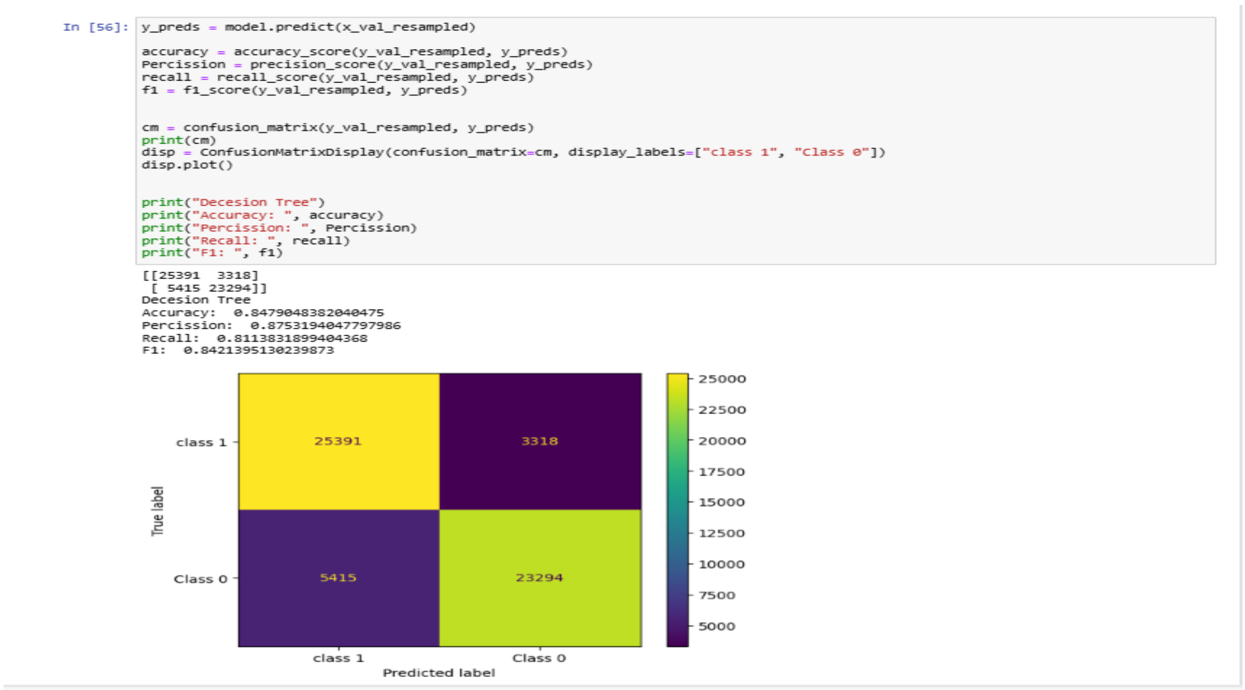
- **Accuracy:** 60.09% — 60% of total predictions were correct.
- **Precision:** 66.2% — about two-thirds of those predicted as cancer actually had cancer.
- **Recall:** 68.1% — the model correctly identified around 68% of actual cancer cases.
- **F1 Score:** 67.1% — balanced performance between detecting true positives and avoiding false alarms.

Conclusion:

- **DT3 achieves a solid balance between recall and precision**, making it a reasonably effective screening model.
- With a **recall of 68.1%**, the model successfully identifies most cancer patients — which is essential in early detection.
- **Precision of 66.2%** means false positives are somewhat controlled, reducing unnecessary anxiety or follow-ups.

4- Model 1.4:

Final one with 100 decision tree with the best result :



```
In [56]: y_preds = model.predict(x_val_resampled)

         accuracy = accuracy_score(y_val_resampled, y_preds)
         Percission = precision_score(y_val_resampled, y_preds)
         recall = recall_score(y_val_resampled, y_preds)
         f1 = f1_score(y_val_resampled, y_preds)

         cm = confusion_matrix(y_val_resampled, y_preds)
         print(cm)
         disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["class 1", "Class 0"])
         disp.plot()

         print("Decesion Tree")
         print("Accuracy: ", accuracy)
         print("Percission: ", Percission)
         print("Recall: ", recall)
         print("F1: ", f1)
```

```
[[25391  3318]
 [ 5415 23294]]
Decesion Tree
Accuracy:  0.8479048382040475
Percission:  0.8753194047797986
Recall:  0.8113831899404368
F1:  0.8421395130239873
```

Confusion Matrix:

- **True Negative (25,391):** Healthy individuals correctly predicted as healthy.
- **False Positive (3,318):** Healthy individuals wrongly predicted as having cancer.
- **False Negative (5,415):** Cancer patients incorrectly predicted as healthy (missed cases).
- **True Positive (23,294):** Cancer patients correctly identified.

Performance Metrics:

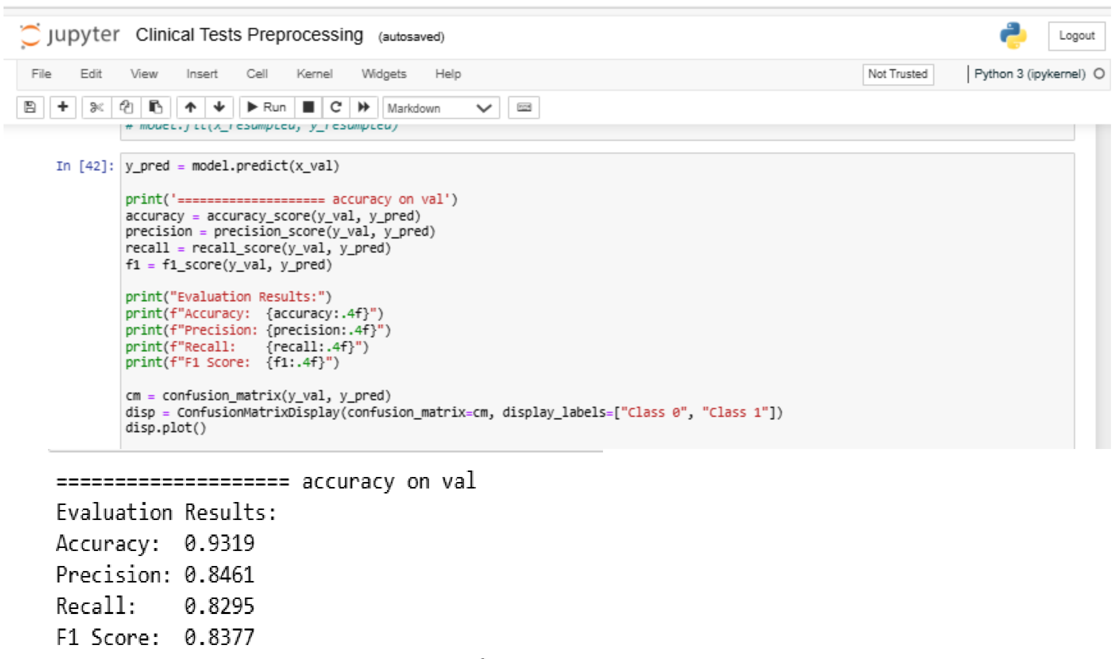
- **Accuracy:** 84.79% — Overall correctness of predictions is high.
- **Precision:** 87.53% — When the model predicts cancer, it is right nearly 88% of the time.
- **Recall:** 81.14% — The model correctly identifies over 81% of all cancer patients.
- **F1 Score:** 84.21% — This harmonic mean of precision and recall shows **strong, balanced performance**.

Conclusion:

- This is the **best-performing Decision Tree model** so far.
- **resampling**, the model achieved:
  - High recall (identifies most cancer cases).
  - High precision (few false alarms).
  - Strong F1 score (balanced and reliable).
- The model makes significantly fewer critical errors (false negatives) compared to DT1, and achieves better overall balance than DT2 and DT3.

Model 2 (Clinical Tests Preprocessing):

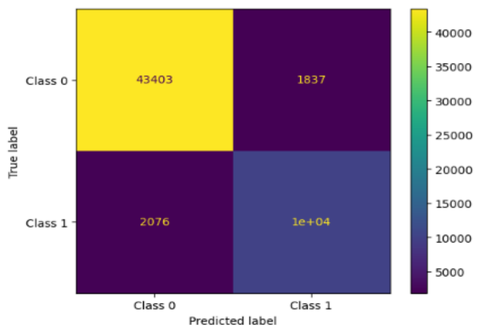
1-Model 2.1



```
# model.fit(x_resampled, y_resampled)

In [42]: y_pred = model.predict(x_val)

         print('==================== accuracy on val')
         accuracy = accuracy_score(y_val, y_pred)
         precision = precision_score(y_val, y_pred)
         recall = recall_score(y_val, y_pred)
         f1 = f1_score(y_val, y_pred)

         print("Evaluation Results:")
         print(f"Accuracy:  {accuracy:.4f}")
         print(f"Precision: {precision:.4f}")
         print(f"Recall:    {recall:.4f}")
         print(f"F1 Score:  {f1:.4f}")

         cm = confusion_matrix(y_val, y_pred)
         disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Class 0", "Class 1"])
         disp.plot()
```

```
==================== accuracy on val
Evaluation Results:
Accuracy:  0.9319
Precision: 0.8461
Recall:    0.8295
F1 Score:  0.8377
```

```
Out[42]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x706712a75bb0>
```



Algorithm Type:

- This model is a **supervised learning classifier** (likely a Decision Tree or similar), applied on clinical data to predict early signs of gastric cancer.
  It uses features derived from **clinical tests**, such as medical lab results or patient examination records. The model was trained and evaluated using a **validation dataset** (x_val, y_val).

Confusion Matrix:

- **True Negatives (43,403):** Healthy individuals correctly predicted as healthy.
- **False Positives (1,837):** Healthy individuals wrongly predicted as having cancer.
- **False Negatives (2,076):** Cancer patients incorrectly predicted as healthy.
- **True Positives (10,000):** Cancer patients correctly identified

Performance Metrics:

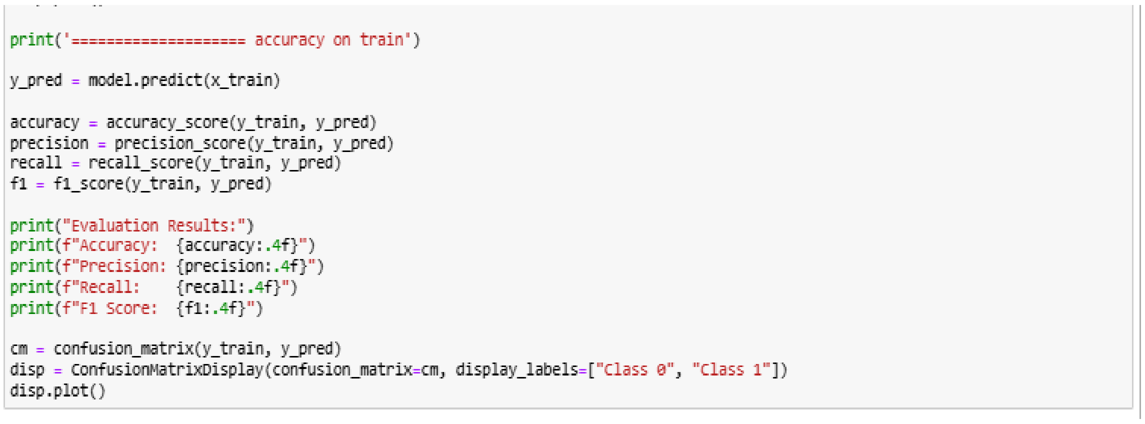
- **Accuracy:** 93.19% — A high percentage of total correct predictions.
- **Precision:** 84.61% — When the model predicts cancer, it is correct about 85% of the time.

- **Recall:** 82.95% — The model successfully detects nearly 83% of all real cancer cases.
- **F1 Score:** 83.77% — A strong balance between precision and recall, showing the model is both accurate and reliable.

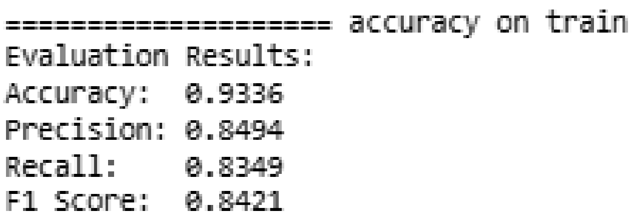
Conclusion:

- This clinical model performs **exceptionally well** in early detection of gastric cancer.
- It maintains a **high recall** (important for minimizing missed cancer cases), while also keeping **false positives relatively low.**

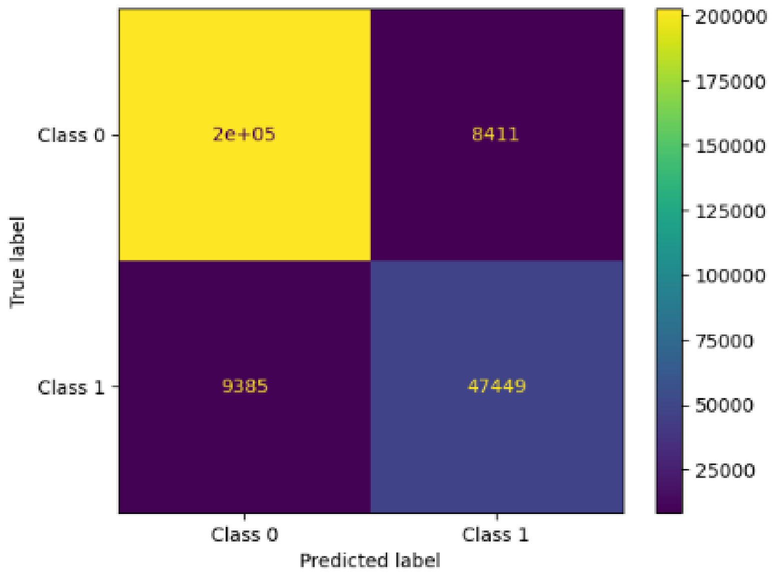
2- Model 2.2



```
print('=================== accuracy on train')

y_pred = model.predict(x_train)

accuracy = accuracy_score(y_train, y_pred)
precision = precision_score(y_train, y_pred)
recall = recall_score(y_train, y_pred)
f1 = f1_score(y_train, y_pred)

print("Evaluation Results:")
print(f"Accuracy:  {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall:    {recall:.4f}")
print(f"F1 Score:  {f1:.4f}")

cm = confusion_matrix(y_train, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Class 0", "Class 1"])
disp.plot()
```



```
==================== accuracy on train
Evaluation Results:
Accuracy:  0.9336
Precision: 0.8494
Recall:    0.8349
F1 Score:  0.8421
```



Algorithm Type:

This is the training evaluation of your clinical model, likely a Decision Tree or similar classifier. It was trained on clinical test data to predict early gastric cancer signs.

Confusion Matrix:

- **True Negatives (200,000):** Healthy patients correctly predicted as healthy.
- **False Positives (8,411):** Healthy patients wrongly predicted as cancer-positive.
- **False Negatives (9,385):** Cancer cases missed by the model.
- **True Positives (47,449):** Cancer cases correctly identified.

Training Performance:

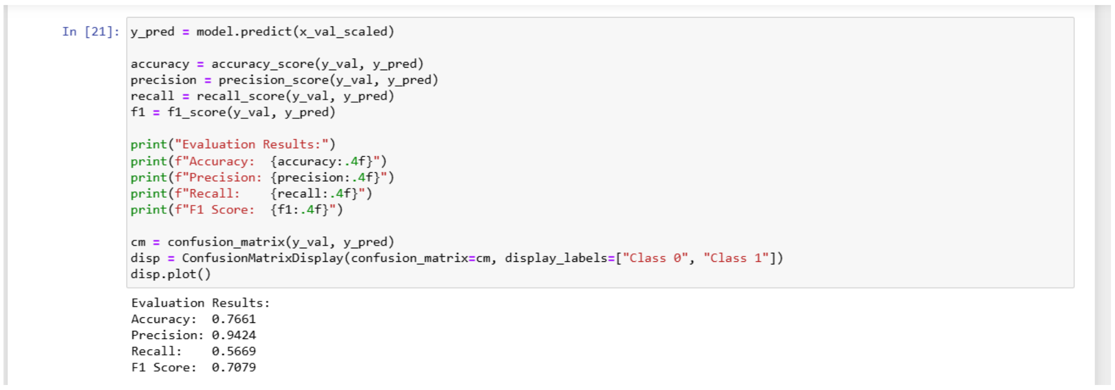
- **Accuracy:** 93.36%.
- **Precision:** 84.94%.
- **Recall:** 83.49%.
- **F1 Score:** 84.21%.

Conclusion:

- The model performs very well on training data, with **high precision and recall**.
- **Overfitting does not appear to be a major issue**, as training and validation scores are closely aligned.
- The model maintains a **good balance**: it detects most cancer cases and keeps false positives at a manageable level.

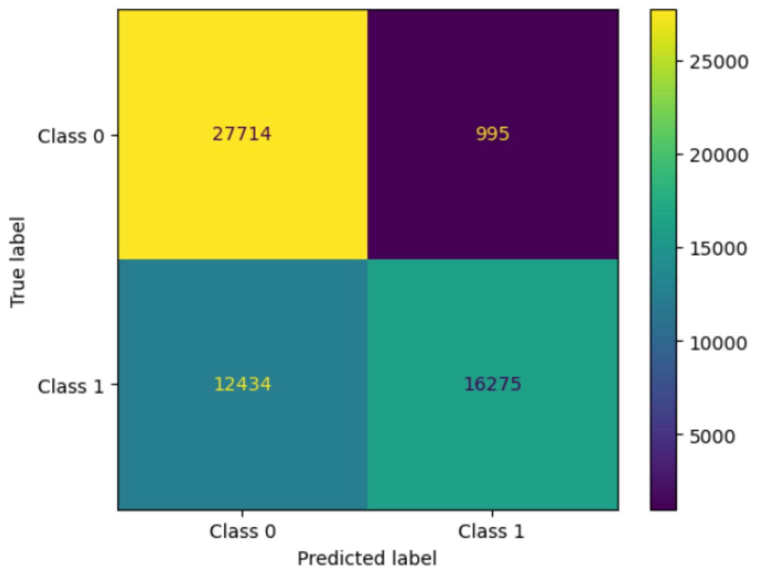
Model 3 (Genetic Test) :

1-Model 3.1 (Scaled Clinical Test Model (Validation Set)



```
In [21]: y_pred = model.predict(x_val_scaled)

         accuracy = accuracy_score(y_val, y_pred)
         precision = precision_score(y_val, y_pred)
         recall = recall_score(y_val, y_pred)
         f1 = f1_score(y_val, y_pred)

         print("Evaluation Results:")
         print(f"Accuracy:  {accuracy:.4f}")
         print(f"Precision: {precision:.4f}")
         print(f"Recall:    {recall:.4f}")
         print(f"F1 Score:  {f1:.4f}")

         cm = confusion_matrix(y_val, y_pred)
         disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Class 0", "Class 1"])
         disp.plot()

         Evaluation Results:
         Accuracy:  0.7661
         Precision: 0.9424
         Recall:    0.5669
         F1 Score:  0.7079
```

```
Out[21]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7dfcc3bcb170>
```



Algorithm :

- This model was trained using scaled clinical data, likely with a **tree-based or logistic model** that benefits from feature scaling (e.g., Logistic Regression, SVM). The goal is to **predict early gastric cancer** based on patient clinical features.

Confusion Matrix (Validation Data):

- **True Negatives (27,714):** Non-cancer cases correctly identified.
- **False Positives (995):** Healthy patients wrongly flagged as cancer-positive.
- **False Negatives (12,434):** Cancer patients missed by the model (a concern).
- **True Positives (16,275):** Cancer patients correctly identified.

Performance Metrics:

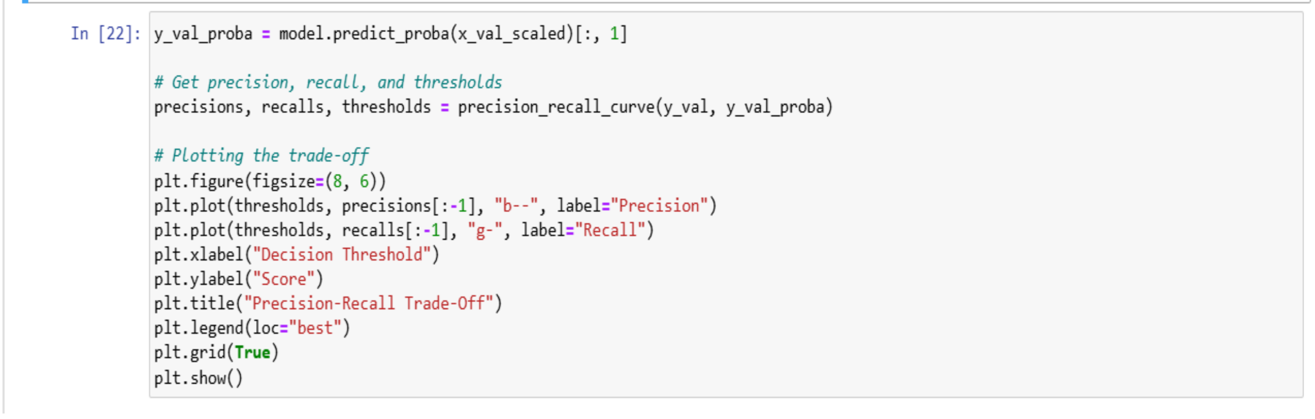
- **Accuracy:** 76.61%
- **Precision:** 94.24%
- **Recall:** 56.69%
- **F1 Score:** 70.79%

Conclusion:

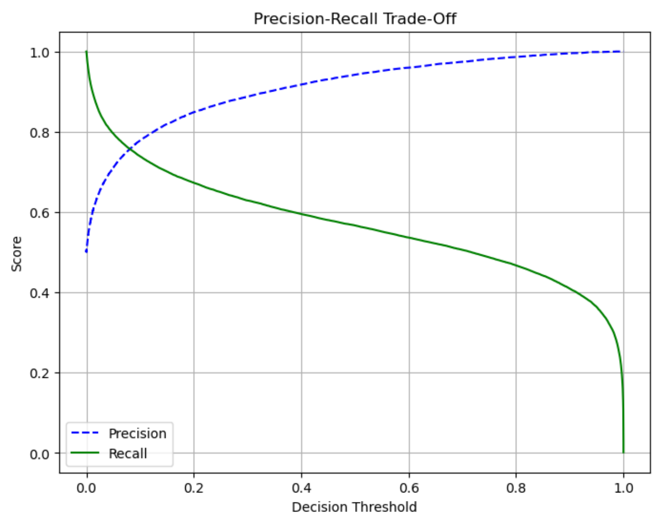
- **High Precision (94%)**: Most predicted cancer cases are indeed cancer, meaning **very few false alarms**.
- **Moderate Recall (57%)**: The model **misses a notable number of cancer cases**, which could be dangerous in a screening setting.
- **Balanced F1 Score (~71%)**: Shows a reasonable trade-off between precision and recall.
- **Acceptable Accuracy (77%)**: Indicates good overall correctness, but must be weighed against the importance of recall in cancer detection.

Analyzing Precision-Recall Trade-Off to Optimize Cancer Detection Thresholds:

1-



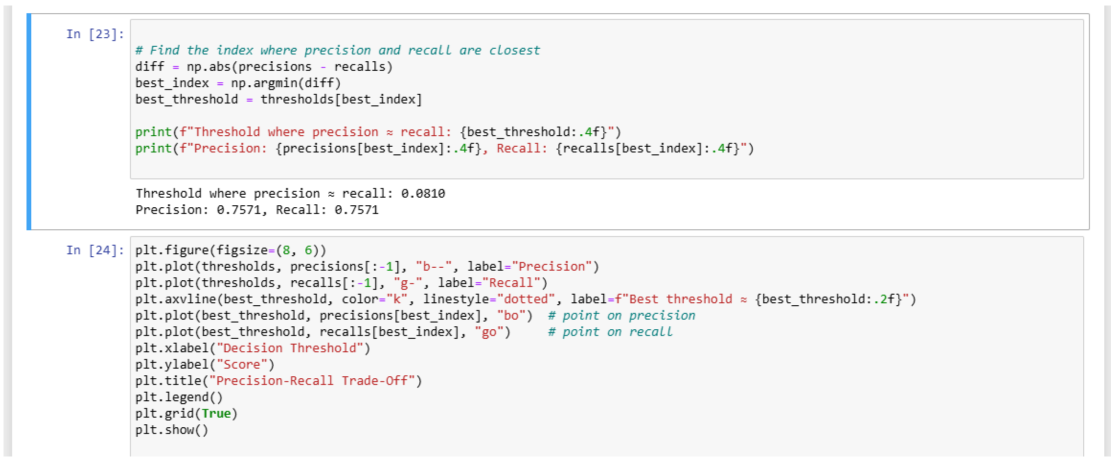
```
In [22]: y_val_proba = model.predict_proba(x_val_scaled)[:, 1]

         # Get precision, recall, and thresholds
         precisions, recalls, thresholds = precision_recall_curve(y_val, y_val_proba)

         # Plotting the trade-off
         plt.figure(figsize=(8, 6))
         plt.plot(thresholds, precisions[:-1], "b--", label="Precision")
         plt.plot(thresholds, recalls[:-1], "g-", label="Recall")
         plt.xlabel("Decision Threshold")
         plt.ylabel("Score")
         plt.title("Precision-Recall Trade-Off")
         plt.legend(loc="best")
         plt.grid(True)
         plt.show()
```



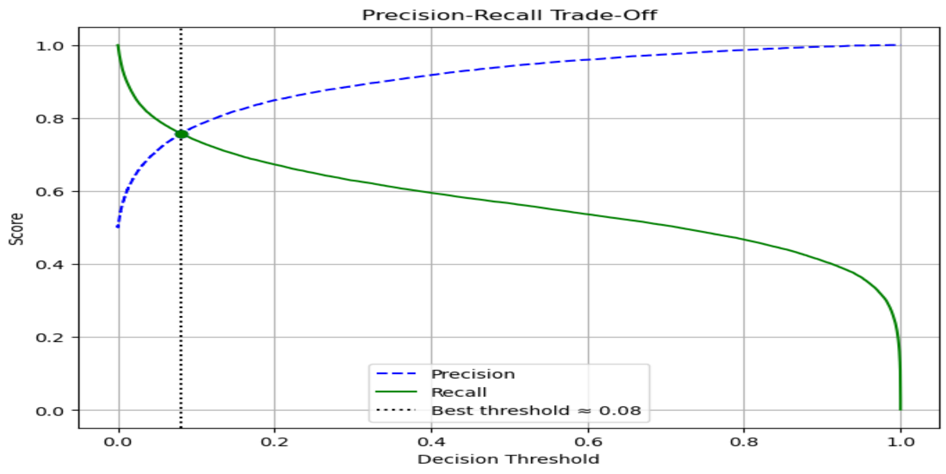
1-Summary (Threshold = 0.08):

- **Balanced Precision & Recall**: Both ≈ **0.76**
- **Moderate Threshold**: Predicts positive at **8% confidence**
- **Best For**: Business decisions needing **few false alarms** (e.g., spam filtering, credit scoring)
- **Pros**: Stable, fewer false positives
- **Cons**: May miss some true positives

2-



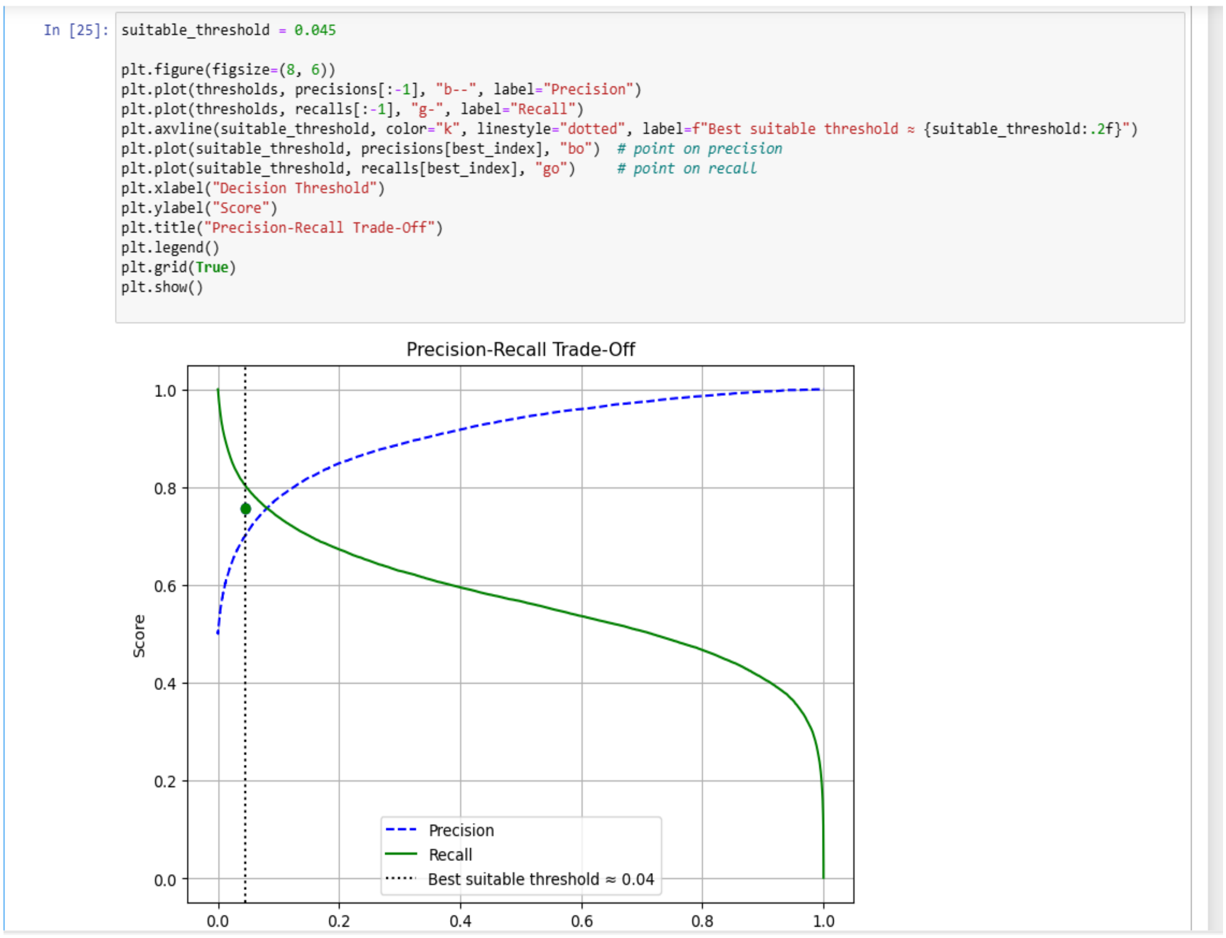
```
In [23]: # Find the index where precision and recall are closest
         diff = np.abs(precisions - recalls)
         best_index = np.argmin(diff)
         best_threshold = thresholds[best_index]

         print(f"Threshold where precision ≈ recall: {best_threshold:.4f}")
         print(f"Precision: {precisions[best_index]:.4f}, Recall: {recalls[best_index]:.4f}")

         Threshold where precision ≈ recall: 0.0810
         Precision: 0.7571, Recall: 0.7571

In [24]: plt.figure(figsize=(8, 6))
         plt.plot(thresholds, precisions[:-1], "b--", label="Precision")
         plt.plot(thresholds, recalls[:-1], "g-", label="Recall")
         plt.axvline(best_threshold, color="k", linestyle="dotted", label=f"Best threshold ≈ {best_threshold:.2f}")
         plt.plot(best_threshold, precisions[best_index], "bo")  # point on precision
         plt.plot(best_threshold, recalls[best_index], "go")     # point on recall
         plt.xlabel("Decision Threshold")
         plt.ylabel("Score")
         plt.title("Precision-Recall Trade-Off")
         plt.legend()
         plt.grid(True)
         plt.show()
```



2-Summary (Threshold = 0.045):

- **Recall-Optimized**: Catches **most positives** (precision ≈ **0.75**)
- **Low Threshold**: Flags positives at just **4.5% confidence**

3-

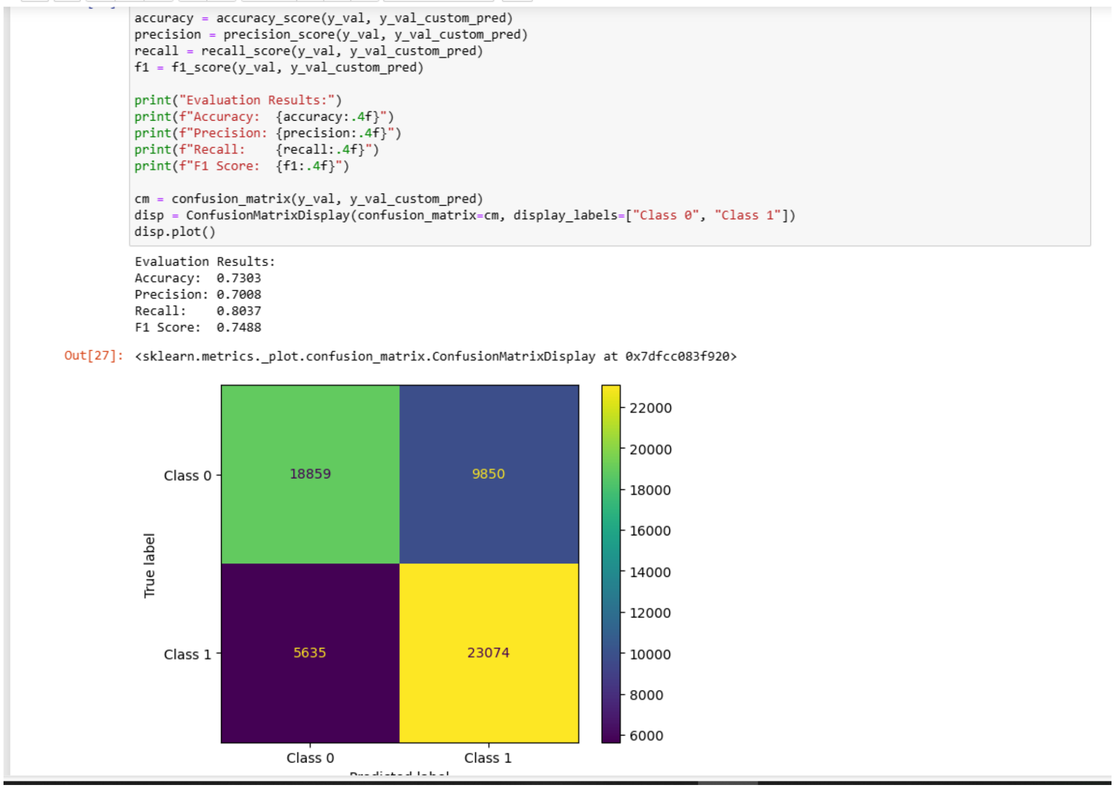


```
In [25]: suitable_threshold = 0.045

         plt.figure(figsize=(8, 6))
         plt.plot(thresholds, precisions[:-1], "b--", label="Precision")
         plt.plot(thresholds, recalls[:-1], "g-", label="Recall")
         plt.axvline(suitable_threshold, color="k", linestyle="dotted", label=f"Best suitable threshold ≈ {suitable_threshold:.2f}")
         plt.plot(suitable_threshold, precisions[best_index], "bo")  # point on precision
         plt.plot(suitable_threshold, recalls[best_index], "go")     # point on recall
         plt.xlabel("Decision Threshold")
         plt.ylabel("Score")
         plt.title("Precision-Recall Trade-Off")
         plt.legend()
         plt.grid(True)
         plt.show()
```

3- Summary(Threshold = 0.04):

- **Extremely Low Threshold**: Maximizes recall but sacrifices precision
- **Likely Use Case**: **Ultra-sensitive detection** (e.g., rare disease screening)

2- model 3.2(Model Evaluation– Clinical Test Classifier):



```
accuracy = accuracy_score(y_val, y_val_custom_pred)
precision = precision_score(y_val, y_val_custom_pred)
recall = recall_score(y_val, y_val_custom_pred)
f1 = f1_score(y_val, y_val_custom_pred)

print("Evaluation Results:")
print(f"Accuracy:  {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall:    {recall:.4f}")
print(f"F1 Score:  {f1:.4f}")

cm = confusion_matrix(y_val, y_val_custom_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Class 0", "Class 1"])
disp.plot()
```

```
Evaluation Results:
Accuracy:  0.7303
Precision: 0.7008
Recall:    0.8037
F1 Score:  0.7488
```

```
Out[27]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7dfcc083f920>
```

Model Type:

- This model is a classification model used to **predict gastric cancer based on clinical test features**. It likely uses scaled inputs (x_val_scaled) from clinical data to identify patients at risk.

Confusion Matrix:

- **True Positives (23,074):** Correctly identified cancer cases.
- **True Negatives (18,859):** Correctly identified non-cancer cases.
- **False Positives (9,850):** Healthy people wrongly flagged as having cancer.
- **False Negatives (5,635):** Cancer cases that the model missed — critical to minimize in healthcare.

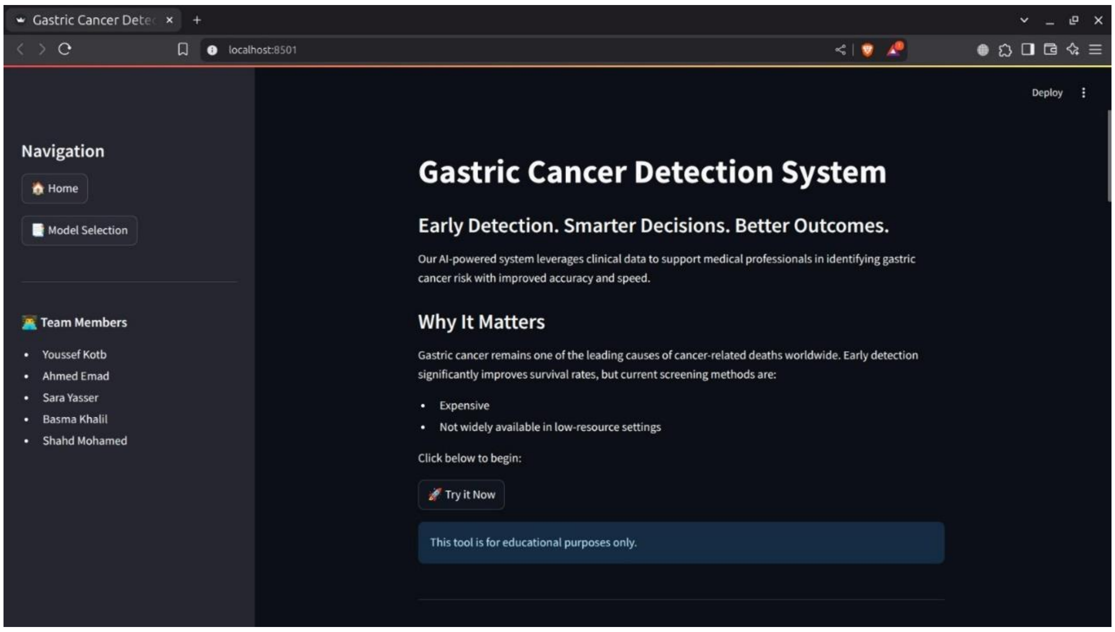
Performance Metrics:

- **Accuracy:** 0.7303 : The model correctly predicts about 73.03% of the total cases.
- **Precision:** 0.7008 : Among all the patients the model predicted as having cancer, ~70.08% actually had cancer.
- **Recall (Sensitivity):** 0.8037: The model correctly identified ~80.37% of actual cancer cases.
- **F1 Score:** 0.7488 : Balanced measure between precision and recall.

Conclusion:

- This clinical-test-based model performs well as a **screening tool**, prioritizing the **identification of true cancer cases** (high recall), which is essential for early detection. Further tuning may reduce false positives without compromising recall.

Deployment



👥 Team Members

- Youssef Kotb
- Ahmed Emad
- Sara Yasser
- Basma Khalil
- Shahd Mohamed

Gastric Cancer Detection System

Early Detection. Smarter Decisions. Better Outcomes.

Our AI-powered system leverages clinical data to support medical professionals in identifying gastric cancer risk with improved accuracy and speed.

Why It Matters

Gastric cancer remains one of the leading causes of cancer-related deaths worldwide. Early detection significantly improves survival rates, but current screening methods are:

- Expensive
- Not widely available in low-resource settings

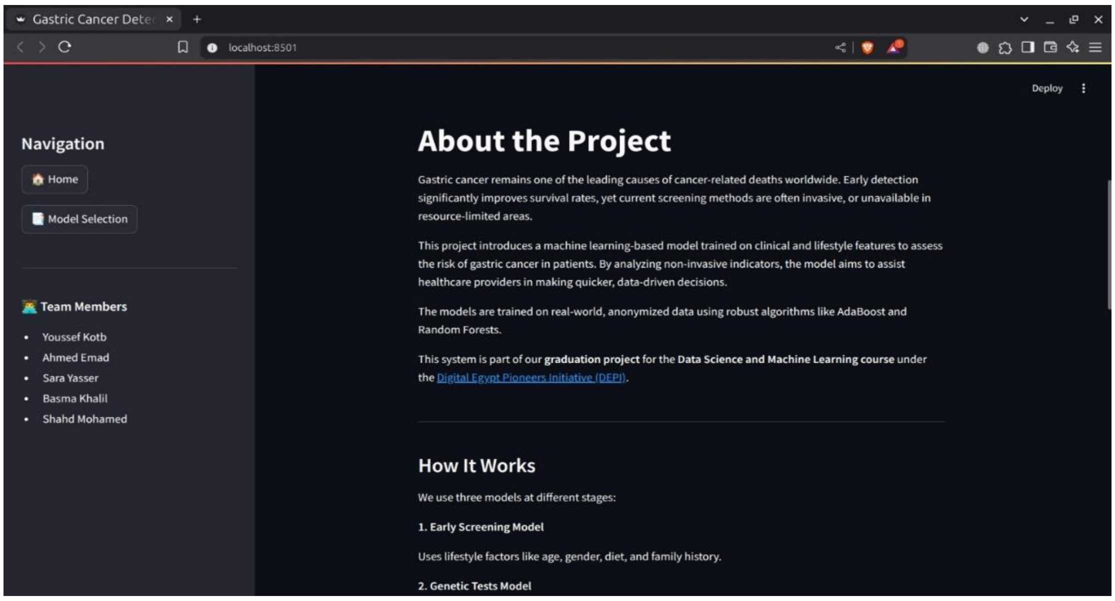
Click below to begin:

🚀 Try it Now

This tool is for educational purposes only.

Early Detection System for Gastric Cancer :

Our AI model has been deployed on an easy-to-use platform, allowing doctors to input clinical data and receive instant risk assessments. The system is designed to be **fast, accurate, and accessible even in low-resource settings**, contributing to improved early diagnosis opportunities.



Team Members

- Youssef Kotb
- Ahmed Emad
- Sara Yasser
- Basma Khalil
- Shahd Mohamed

About the Project

Gastric cancer remains one of the leading causes of cancer-related deaths worldwide. Early detection significantly improves survival rates, yet current screening methods are often invasive, or unavailable in resource-limited areas.

This project introduces a machine learning-based model trained on clinical and lifestyle features to assess the risk of gastric cancer in patients. By analyzing non-invasive indicators, the model aims to assist healthcare providers in making quicker, data-driven decisions.

The models are trained on real-world, anonymized data using robust algorithms like AdaBoost and Random Forests.

This system is part of our **graduation project** for the **Data Science and Machine Learning course** under the Digital Egypt Pioneers Initiative (DEPI).

How It Works

We use three models at different stages:

1. Early Screening Model

Uses lifestyle factors like age, gender, diet, and family history.

2. Genetic Tests Model

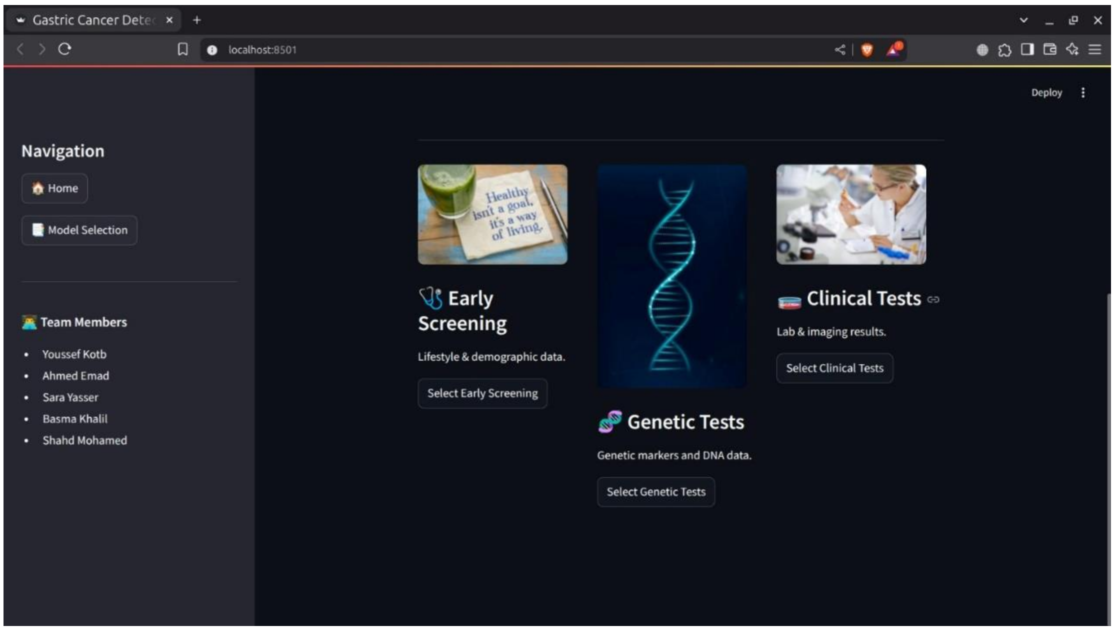
Gastric Cancer Risk Assessment System

- Our team has developed and deployed a multi-stage AI screening system for gastric cancer risk evaluation. The platform features:

1. **User-Friendly Interface**
- Clear navigation (Home/Model Selection/Team)

2. **Technical Implementation**
- Two specialized ML models deployed:
    - **Early Screening Model**: Analyzes lifestyle factors (age, diet, family history)
    - **Genetic Tests Model**: Processes biomarker data (currently in development)

3. **Clinical Value Proposition**
- Provides non-invasive preliminary assessment
- Generates instant risk scores with explanation
- Designed for use in low-resource settings



Gastric Cancer Assessment System - Multi-Tier Screening Interface

Our system offers three specialized screening pathways tailored to available data types:

1. **Early Screening**
- Input demographic & lifestyle factors (age, gender, dietary habits)
- Generates instant preliminary risk assessment
- Ideal for initial evaluation before clinical testing
2. **Clinical Tests**
- Processes lab results and imaging data
- Delivers enhanced diagnostic accuracy
- Designed for equipped medical facilities
3. **Genetic Tests**
- Specialized genomic marker analysis (currently in development)
- Targets patients with family history
- Works with DNA sequencing data

"A comprehensive platform providing progressive diagnosis - from basic screening to advanced biomarker analysis"

This interface serves diverse user needs:

- Patients: Accessible self-assessment starting point
- Physicians: Tiered analysis matching available clinical data
- Researchers: Future genomic model development tracking