Final Project Report Template

1. Introduction

1.1. Project overviews

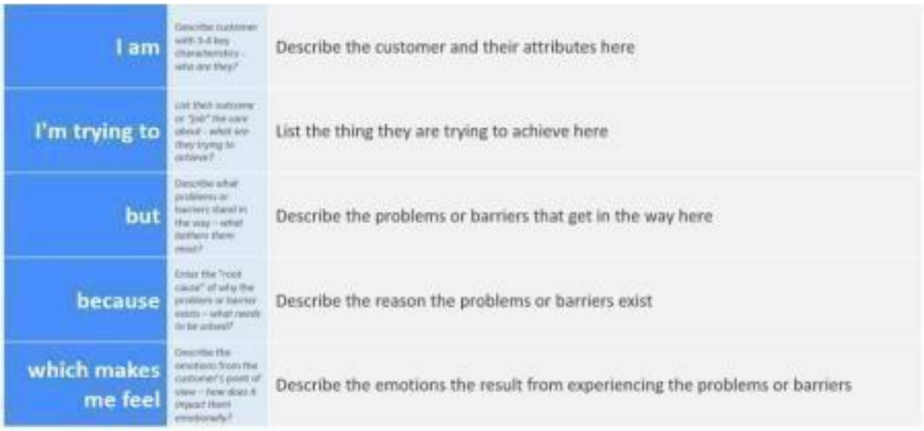
The goal of this project is to develop a machine learning model that will enable the classification of thyroid disorders using patient data from a dataset.

1.2. Objectives

- Clean and prepare the dataset.
- Train and test Random Forest, XGBoost, and SVC models.
- Evaluate model performance using accuracy and other metrics.
- Select the most significant features.
- Choose the most accurate model for predicting thyroid conditions.
- Deploy the best model in a Flask web application with home, predict, and submit pages.

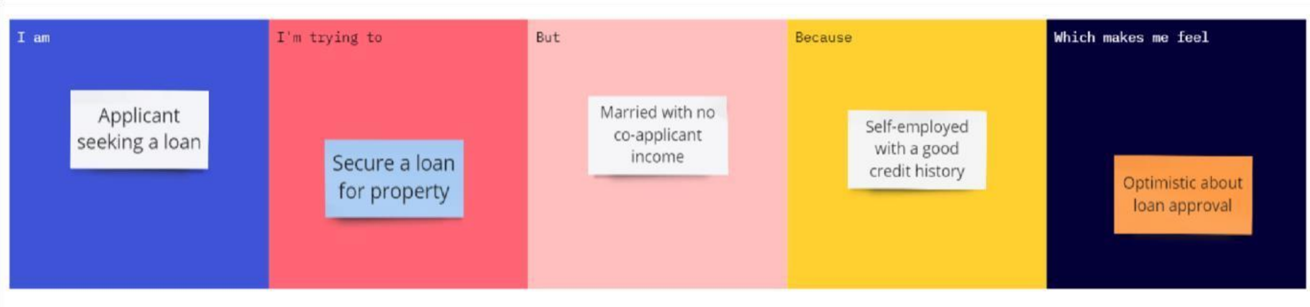
2. Project Initialization and Planning Phase 2.1.Define Problem Statement Define Problem Statements (Customer Problem Statement Template):

Create a problem statement to understand your customer's point of view. The Customer Problem Statement template helps you focus on what matters to create experiences people will love. A well-articulated customer problem statement allows you and your team to find the ideal solution for your customers' challenges. Throughout the process, you'll also be able to empathize with your customers, which helps you better understand how they perceive your product or service.



I am	Describe customer with 3-4 key characteristics - who are they?	Describe the customer and their attributes here
I'm trying to	List their outcome or "job" the care about - what are they trying to achieve?	List the thing they are trying to achieve here
but	Describe what problems or barriers stand in the way – what bothers them most?	Describe the problems or barriers that get in the way here
because	Enter the "root cause" of why the problem or barrier exists – what needs to be solved?	Describe the reason the problems or barriers exist
which makes me feel	Describe the emotions from the customer's point of view – how does it impact them emotionally?	Describe the emotions the result from experiencing the problems or barriers

Reference: https://miro.com/templates/customer-problem-statement/ **Example:**



I am	I'm trying to	But	Because	Which makes me feel
Applicant seeking a loan	Secure a loan for property	Married with no co-applicant income	Self-employed with a good credit history	Optimistic about loan approval

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	A healthcare provider	Diagnose thyroid conditions quickly	Need to ensure accuracy	Early and accurate diagnosis can improve patient outcomes	Anxious about potential misdiagnosis
PS-2	A data scientist	Build a robust ML model for thyroid classification	Have imbalanced dataset	Some thyroid conditions are rare	Challenged by the need to balance precision and recall
PS-3	A hospital administrator	Implement an automated diagnostic tool	Have budget constraints	Automation can reduce operational costs and improve efficiency	Worried about initial investment and ROI
PS-4	A clinical researcher	Improve the accuracy of thyroid condition classification	Have a limited dataset	Accurate models can lead to better research outcomes	Determined but concerned about data limitations

PS-5	A patient	Understand my thyroid condition better	The technical details are complex	Better understanding can lead to more informed decisions about my health	Confused about the medical jargon and results

2. Project Proposal (Proposed Solution)

Project Proposal (Proposed Solution) template

The proposal study seeks to improve accuracy and efficiency in thyroid classification by utilizing machine learning. It addresses inefficiencies in the system and promises improved operations, lower risks, and happier patient's. Real-time decision-making and a credit model driven by machine learning are important characteristics.

Project Overview	
Objective	The primary objective is to have knowledge on pre-processing the data/transformation techniques on outlier and some visualization concepts and Know fundamental concepts and techniques used for machine learning in thyroid classification.
Scope	The study uses machine learning to create a more reliable and effective method, thoroughly evaluating and improving the thyroid classification procedure.

Problem Statement	
Description	Addressing inaccuracies and inefficiencies in the current thyroid classification system adversely affects operational efficiency and customer satisfaction.
Impact	Solving these issues will result in improved operational efficiency, reduced risks, and an overall enhancement in the lending process, contributing to patient satisfaction and organizational success.
Proposed Solution	
Approach	Employing machine learning techniques to analyze and predict creditworthiness, creating a dynamic and adaptable thyroid classification.
Key Features	Implementation of a machine learning-based credit assessment model

Resource Requirements

Resource Requirements

Resource Type	**Description**	**Specification/Allocation**
Hardware		

Computing Resources	CPU/GPU specifications, number of cores	T4 GPU
Memory	RAM specifications	8 GB
Storage	Disk space for data, models, and logs	1 TB SSD
Software		
Frameworks	Python frameworks	Flask
Libraries	Additional libraries	scikit-learn, pandas, NumPy, matplotlib
Development Environment	IDE, version control	Jupyter Notebook, Git
Data		
Data	Source, size, format	Kaggle dataset, 614, csv UCI dataset, 690, csv

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-3	Data Preprocessing	TC-10	Splitting data into train and test	1	Medium	Surya
Sprint-3	Data Preprocessing	TC-11	Handling Imbalanced data	1	Low	Sai Abhinay
Sprint-3	Data Preprocessing	TC-12	Applying StandardScaler	2	High	Jaswanth
Sprint-4	Model building	TC-13	Random forest model	2	Medium	Surya
Sprint-4	Model building on selected columns	TC-14	Random forest classifier model	2	Medium	Rohith
Sprint-4	Model building on selected columns	TC-15	XGBClassifier model	1	Low	Jaswanth
Sprint-4	Model building on selected columns	TC-16	SVC model	1	Medium	Sai Abhinay
Sprint-4	Model building on selected columns	TC-17	Evaluating performance of the model using grid search CV and Saving the model	2	Low	Surya

Sprint-5	Application Building	TC-18	Building Html pages	1	High	Sai Abhinay

2.3 project planning template Use the below template to create a product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data Collection	TC-1	Download the dataset	2	High	Jaswanth
Sprint-2	Visualising and Analysing Data	TC-2	Importing the libraries	1	High	Rohith
Sprint-2	Visualising and Analysing Data	TC-3	Read the dataset	1	Low	Sai Abhinay
Sprint-3	Data Preprocessing	TC-4	Checking for null values	2	Low	Surya
Sprint-3	Data Preprocessing	TC-5	Descriptive analysis	1	Medium	Rohith
Sprint-3	Data Preprocessing	TC-6	Splitting the data X and Y	1	Medium	Jaswanth
Sprint-3	Data Preprocessing	TC-7	Converting the Datatype	1	High	Surya
Sprint-3	Data Preprocessing	TC-8	Handling categorical values	1	Low	Sai Abhinay
Sprint-3	Data Preprocessing	TC-9	Checking correlation	2	Medium	Rohith
Sprint-5	Application Building	TC-19	Build Flask code	2	High	Jaswanth
Sprint-5	Application Building	TC-20	Run the application	2	High	Rohith

3. Data Collection and Preprocessing Phase

3.1 Data Collection Plan and Raw Data Sources Identified

Section	Description
Project Overview	The Thyroid Classification project aims to develop a robust machine learning model to accurately classify various thyroid disorders such as hyperthyroidism, hypothyroidism, and thyroid cancer. By leveraging diverse and meticulously curated data sources, including clinical databases, public health datasets, and electronic health records, the project ensures high data quality and integrity. The ultimate goal is to provide healthcare professionals with a reliable diagnostic tool, enhancing patient outcomes through precise and timely diagnosis. Ethical considerations and data protection compliance are integral to the project's data strategy
Data Collection Plan	**Thyroid Disease Dataset:** • **Description:** A publicly available dataset containing records of patients with various thyroid conditions. • **Data Types:** Numerical (T3, T4, TSH levels), Categorical (gender, diagnosis). • **Access:**https://www.kaggle.com/datasets/emmanuelfwerr/thy roiddisease-data/code
Raw Data Sources Identified	**Thyroid Disease Dataset**

Raw data sources template

Source Name	Description	Location/URL	Format	Size	Access Permissions
Dataset 1	This dataset likely includes anonymized medical records of patients diagnosed with thyroid diseases. It typically includes features such as patient demographics (age, gender), clinical measurements (T3, T4, TSH levels), thyroid function test results, and possibly other relevant medical history data	https://www.kaggle.com/datasets/emmanuelfwerr/thyroid-diseasedata/code	CSV	732 KB	Public

3.2 Data quality Template

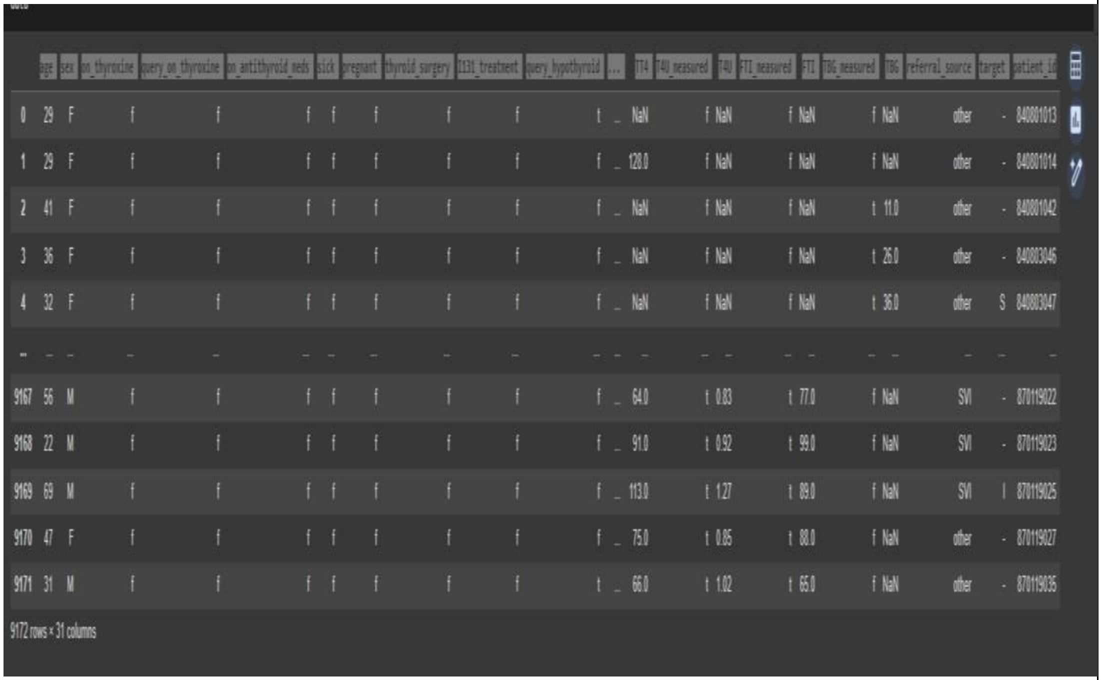
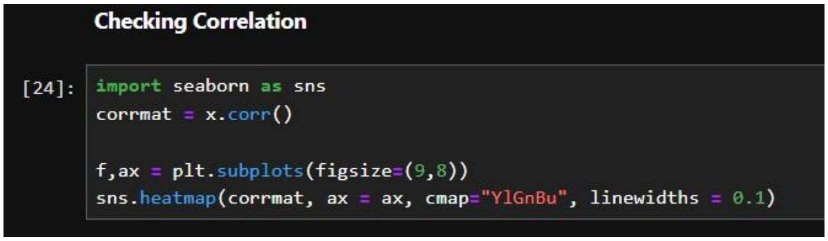
Data Source	Data Quality Issue	Severity	Resolution Plan
Kaggle Dataset	Missing values are 'TSH_measured','T3_measured','TT4_measured','T4U_measured','FTI_measured','TBG_measured','referral_source' and 'patient_id'	Moderate	Use mean/median imputation.
Kaggle Dataset	Categorical data in the dataset	Moderate	Encoding has to be done in the data.

3.3 Data Exploration and Preprocessig Template

Data Exploration and Preprocessing Template

Identifies data sources, assesses quality issues like missing values and duplicates, andimplements resolution plans to ensure accurate and reliable analysis.

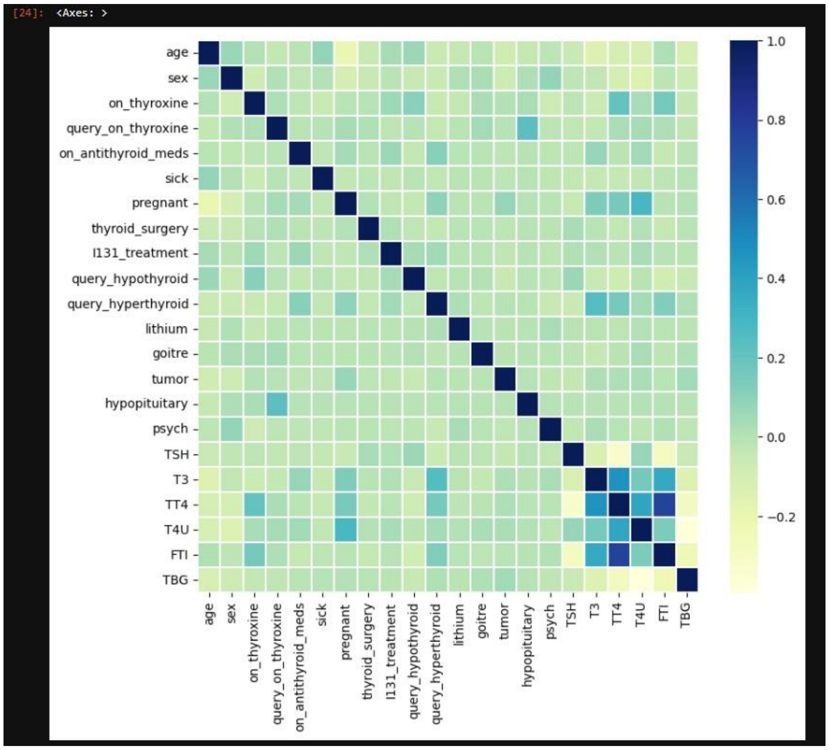
Section	**Description**
Data Overview	For our thyroid classification project, we analyzed a dataset with 9,172 rows and 31 columns, including attributes like age, sex, thyroid medication details, and various thyroid test results. We started by checking for null values and performing a descriptive analysis to understand the data distribution. The dataset was split into features (X) and the target variable (Y). Data types were converted, categorical values were handled, and correlations were checked. We addressed data imbalance and standardized the features using StandardScaler. We built models using RandomForestClassifier,XGBClassifier, and SVC, applying hyperparameter tuning for optimization. Our evaluation through classification reports, confusion matrices, and accuracy scores revealed that XGBClassifier achieved the highest accuracy, making it the best performing model among the three.

	
Univariate Analysis	we performed univariate analysis to understand the distribution and characteristics of individual features in the dataset. This involved examining each feature's summary statistics, such as mean, median, mode, standard deviation, and range, to identify central tendencies and variability. For categorical variables like sex, on_thyroxine, and referral_source, we calculated frequency distributions to observe the most common categories. For continuous variables such as age, TT4, T4U, and FTI, we plotted histograms and box plots to visualize their distributions and detect any outliers or skewness. This analysis provided insights into the underlying patterns and variability of each feature, guiding further steps in data preprocessing and model building.
Bivariate Analysis	

```
      age sex on_thyroxine query_on_thyroxine on_antithyroid_meds sick pregnant thyroid_surgery I131_treatment query_hypothyroid ... TT4 T4U_measured T4U FTI_measured FTI TBG_measured TBG referral_source target patient_id
0     29  F   f            f                  f                   f    f        f               f              t                 ... NaN   f          NaN f            NaN f            NaN other           -      840801013
1     29  F   f            f                  f                   f    f        f               f              f                 ... 128.0 f          NaN f            NaN f            NaN other           -      840801014
2     41  F   f            f                  f                   f    f        f               f              f                 ... NaN   f          NaN f            NaN t            11.0 other          -      840801042
3     36  F   f            f                  f                   f    f        f               f              f                 ... NaN   f          NaN f            NaN t            26.0 other          -      840803046
4     32  F   f            f                  f                   f    f        f               f              f                 ... NaN   f          NaN f            NaN t            36.0 other          S      840803047
...   ... ... ...          ...                ...                 ...  ...      ...             ...            ...               ... ...   ...        ... ...          ... ...          ...  ...            ...    ...
9167  56  M   f            f                  f                   f    f        f               f              f                 ... 64.0  t          0.83 t           77.0 f           NaN SVI             -      870119022
9168  22  M   f            f                  f                   f    f        f               f              f                 ... 91.0  t          0.92 t           99.0 f           NaN SVI             -      870119023
9169  69  M   f            f                  f                   f    f        f               f              f                 ... 113.0 t          1.27 t           89.0 f           NaN SVI             I      870119025
9170  47  F   f            f                  f                   f    f        f               f              f                 ... 75.0  t          0.85 t           80.0 f           NaN other           -      870119027
9171  31  M   f            f                  f                   f    f        f               f              f                 ... 66.0  t          1.02 t           65.0 f           NaN other           -      870119035
9172 rows × 31 columns
```

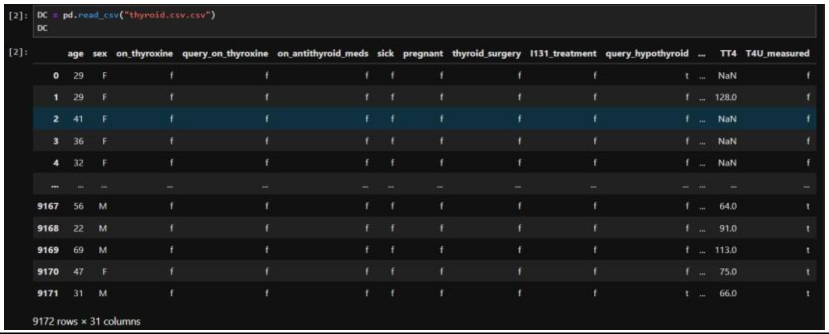
Checking Correlation

```
[24]: import seaborn as sns
      corrmat = x.corr()

      f,ax = plt.subplots(figsize=(9,8))
      sns.heatmap(corrmat, ax = ax, cmap="YlGnBu", linewidths = 0.1)
```

Multivariate Analysis	 [24]: <Axes: >
Outliers and Anomalies	-

Data Preprocessing Code Screenshots

Loading Data



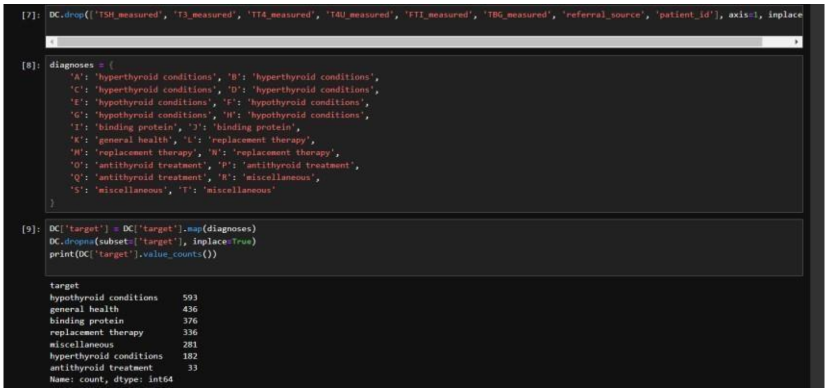
```
[2]: DC = pd.read_csv("thyroid.csv.csv")
     DC
```

[2]:

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_meds	sick	pregnant	thyroid_surgery	I131_treatment	query_hypothyroid	...	TT4	T4U_measured
0	29	F	f	f	f	f	f	f	f	t	...	NaN	f
1	29	F	f	f	f	f	f	f	f	f	...	128.0	f
2	41	F	f	f	f	f	f	f	f	f	...	NaN	f
3	36	F	f	f	f	f	f	f	f	f	...	NaN	f
4	32	F	f	f	f	f	f	f	f	f	...	NaN	f
...	...	...	...	...	...	...	...	...	...	...	...	...	...
9167	56	M	f	f	f	f	f	f	f	f	...	64.0	t
9168	22	M	f	f	f	f	f	f	f	f	...	91.0	t
9169	69	M	f	f	f	f	f	f	f	f	...	113.0	t
9170	47	F	f	f	f	f	f	f	f	f	...	75.0	t
9171	31	M	f	f	f	f	f	f	f	t	...	66.0	t

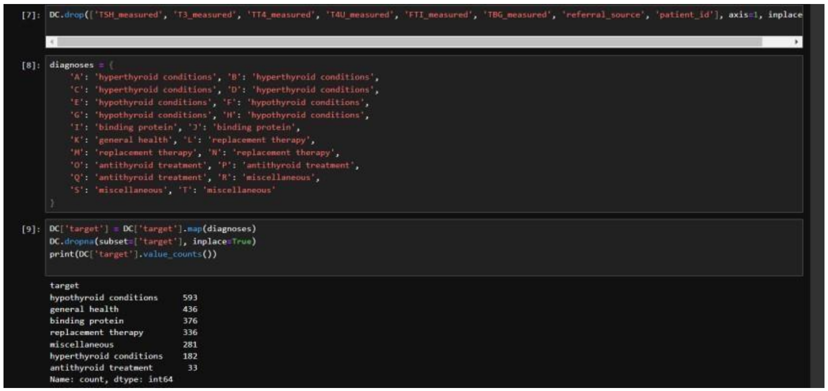
9172 rows × 31 columns

Handling Missing Data



```
[7]: DC.drop(['TSH_measured', 'T3_measured', 'TT4_measured', 'T4U_measured', 'FTI_measured', 'TBG_measured', 'referral_source', 'patient_id'], axis=1, inplace

[8]: diagnoses = {
         'A': 'hyperthyroid conditions', 'B': 'hyperthyroid conditions',
         'C': 'hyperthyroid conditions', 'D': 'hyperthyroid conditions',
         'E': 'hypothyroid conditions', 'F': 'hypothyroid conditions',
         'G': 'hypothyroid conditions', 'H': 'hypothyroid conditions',
         'I': 'binding protein', 'J': 'binding protein',
         'K': 'general health', 'L': 'replacement therapy',
         'M': 'replacement therapy', 'N': 'replacement therapy',
         'O': 'antithyroid treatment', 'P': 'antithyroid treatment',
         'Q': 'antithyroid treatment', 'R': 'miscellaneous',
         'S': 'miscellaneous', 'T': 'miscellaneous'
     }

[9]: DC['target'] = DC['target'].map(diagnoses)
     DC.dropna(subset=['target'], inplace=True)
     print(DC['target'].value_counts())

     target
     hypothyroid conditions     593
     general health             436
     binding protein            376
     replacement therapy        336
     miscellaneous              281
     hyperthyroid conditions    182
     antithyroid treatment       33
     Name: count, dtype: int64
```

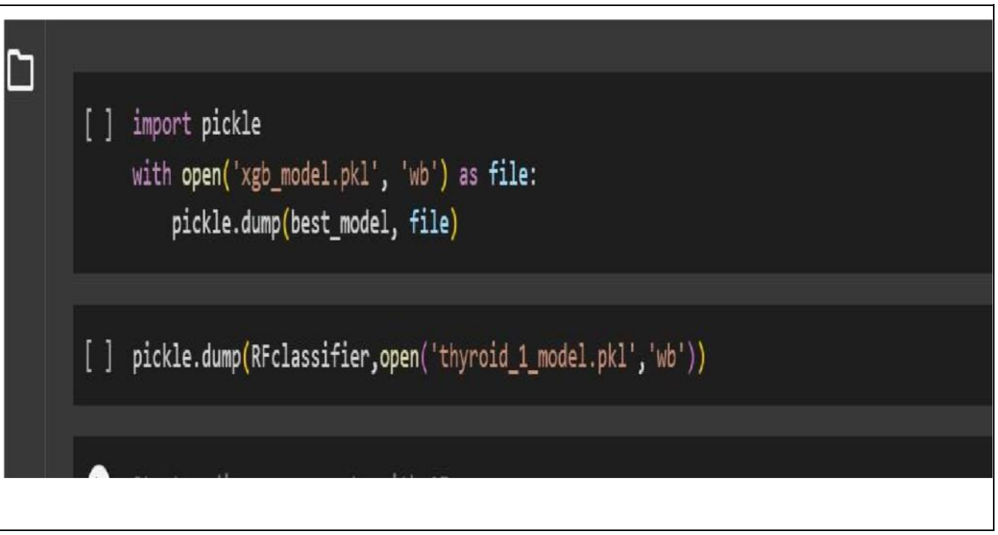
Data Transformation



```
[7]: DC.drop(['TSH_measured', 'T3_measured', 'TT4_measured', 'T4U_measured', 'FTI_measured', 'TBG_measured', 'referral_source', 'patient_id'], axis=1, inplace

[8]: diagnoses = {
         'A': 'hyperthyroid conditions', 'B': 'hyperthyroid conditions',
         'C': 'hyperthyroid conditions', 'D': 'hyperthyroid conditions',
         'E': 'hypothyroid conditions', 'F': 'hypothyroid conditions',
         'G': 'hypothyroid conditions', 'H': 'hypothyroid conditions',
         'I': 'binding protein', 'J': 'binding protein',
         'K': 'general health', 'L': 'replacement therapy',
         'M': 'replacement therapy', 'N': 'replacement therapy',
         'O': 'antithyroid treatment', 'P': 'antithyroid treatment',
         'Q': 'antithyroid treatment', 'R': 'miscellaneous',
         'S': 'miscellaneous', 'T': 'miscellaneous'
     }

[9]: DC['target'] = DC['target'].map(diagnoses)
     DC.dropna(subset=['target'], inplace=True)
     print(DC['target'].value_counts())

     target
     hypothyroid conditions     593
     general health             436
     binding protein            376
     replacement therapy        336
     miscellaneous              281
     hyperthyroid conditions    182
     antithyroid treatment       33
     Name: count, dtype: int64
```

Feature Engineering

Attached the codes in final submission.

Save Processed Data	 [ ] import pickle with open('xgb_model.pkl', 'wb') as file: pickle.dump(best_model, file) [ ] pickle.dump(RFclassifier,open('thyroid_1_model.pkl','wb'))

4. Model Development Phase

4.1 Feature Selection Report Template

In the forthcoming update, each feature will be accompanied by a brief description. Users will indicate whether it's selected or not, providing reasoning for their decision. This process will streamline decision-making and enhance transparency in feature selection.

Feature	Description	Selected (Yes/No)	Reasoning
query_hypothyroid	Query on hypothyroidism	Yes	Relevant for identifying potential cases of hypothyroidism.

query_hyperthyroid	Query on hyperthyroidism	Yes	Relevant for identifying potential cases of hyperthyroidism.
lithium	Lithium medication usage	**Yes**	Relevant as lithium can affect thyroid function.
Tumor	Presence of thyroid tumor	Yes	Tumors can affect thyroid function and are relevant.
hypopituitary	Hypopituitary condition	Yes	Relevant as pituitary disorders can affect thyroid function.
psych	Psychological medication usage	Yes	Relevant as certain psych medications can impact thyroid function.
T3	Triiodothyronine level	Yes	Active thyroid hormone level important for classification.
TSH	Thyroid Stimulating Hormone level	Yes	Primary indicator of thyroid function.

TBG	Thyroxine-Binding Globulin level	Yes	Protein carrier for thyroid hormones relevant for classification.

4.2 Model Selection Report

Model Selection Report

In the forthcoming Model Selection Report, various models will be outlined, detailing their descriptions, hyperparameters, and performance metrics, including Accuracy or F1 Score. This comprehensive report will provide insights into the chosen models and their effectiveness.

Model	Description	Hyperparameters	Performance Metric (e.g., Accuracy, F1 Score)
Random Forest	Ensemble learning method using multiple decision trees for classification.	- Number of trees: 100 - Maximum depth of trees: 10 - Minimum samples per split: 20	Accuracy
	Algorithm that finds a hyperplane to separate classes with the maximum margin.		

Support Vector Machine (SVM)		- Kernel: Linear - Cost (C): 1.0 (controls penalty for misclassification) - Gamma: 0.1 (influences kernel function behavior)	Precision

XGBoost	Gradient boosting framework that trains multiple decision trees sequentially.	- Learning rate: 0.1 (controls step size in weight updates) - Maximum depth of trees: 5 - Minimum samples per leaf: 10	Recall
Logistic Regression	Statistical method that predicts the probability of a binary outcome.	- Regularization parameter (lambda): 0.01 (controls model complexity)	F1 Score (harmonic mean of precision and recall)

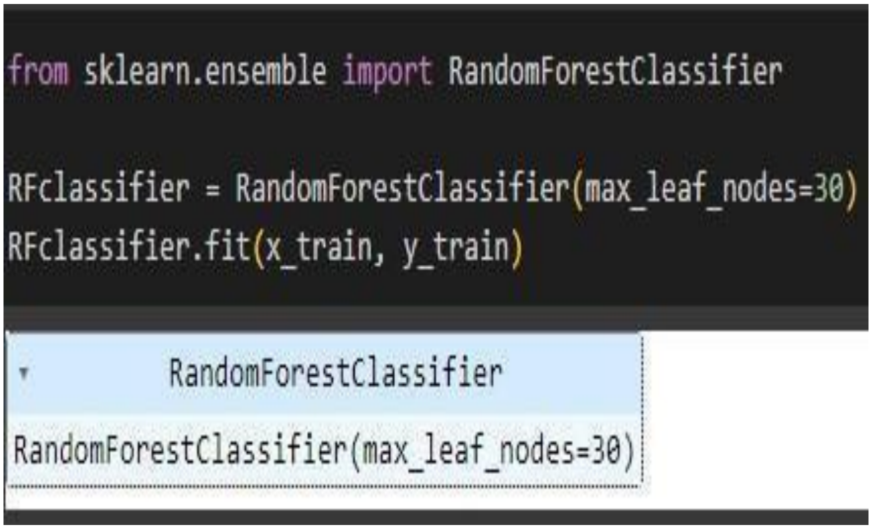
K-Nearest Neighbors (KNN)	Classifies data points based on the majority vote of its k nearest neighbors.	- Number of neighbors (k): 5	AUC-ROC (Area Under the Receiver Operating Characteristic Curve)

4.3 Initial model Training code , Model validation and evaluation report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots

Initial Model Training Code:

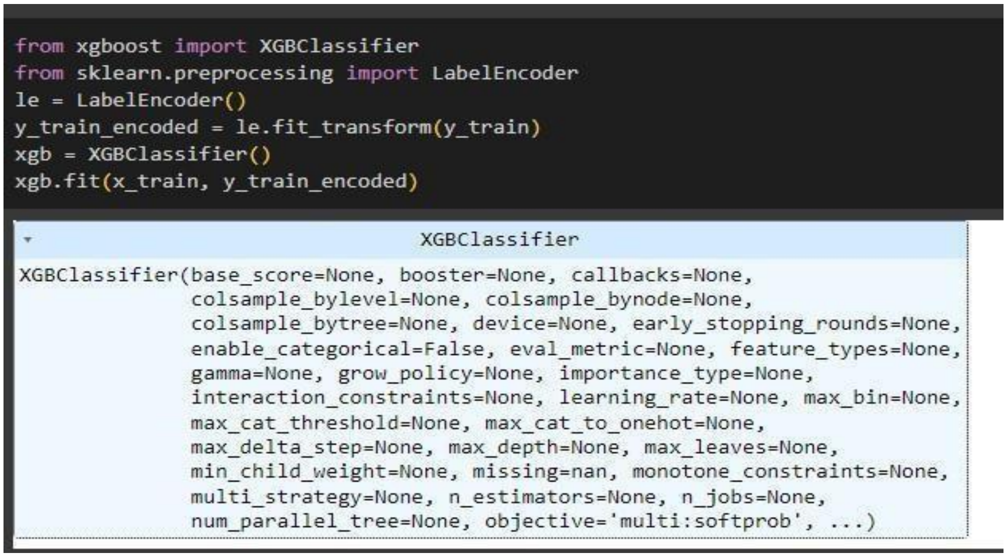
RandomForestClassifier



```
from sklearn.ensemble import RandomForestClassifier

RFclassifier = RandomForestClassifier(max_leaf_nodes=30)
RFclassifier.fit(x_train, y_train)
```

```
▾    RandomForestClassifier
RandomForestClassifier(max_leaf_nodes=30)
```

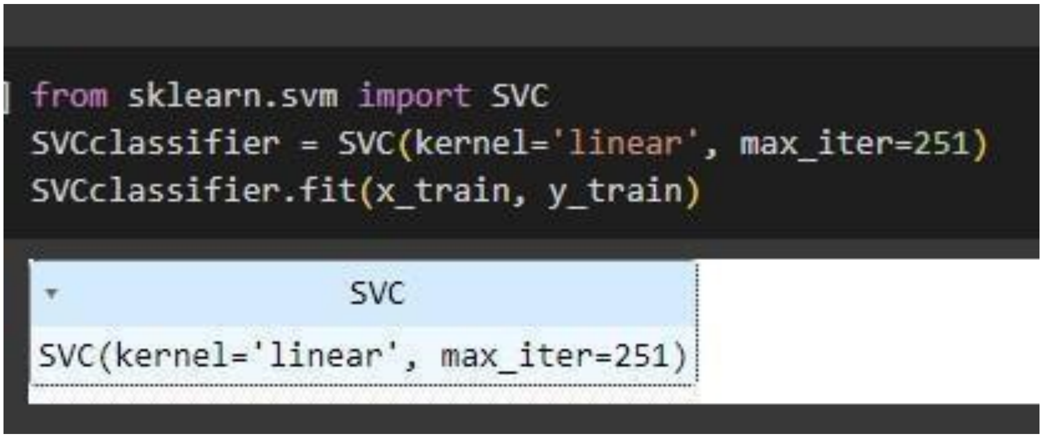
XGB Classifier



```
from xgboost import XGBClassifier
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y_train_encoded = le.fit_transform(y_train)
xgb = XGBClassifier()
xgb.fit(x_train, y_train_encoded)
```

```
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, objective='multi:softprob', ...)
```

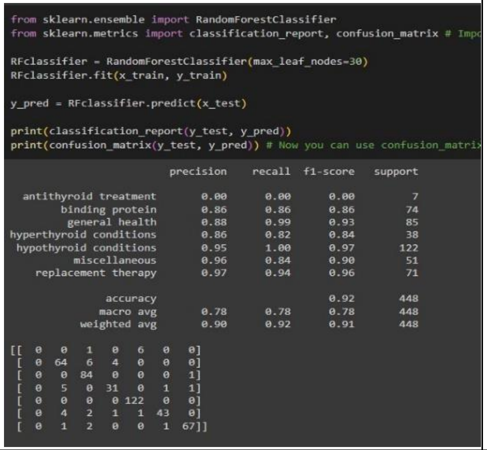
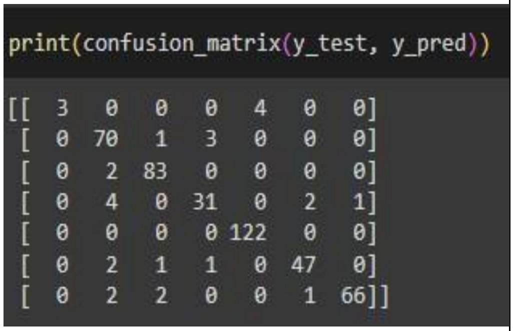
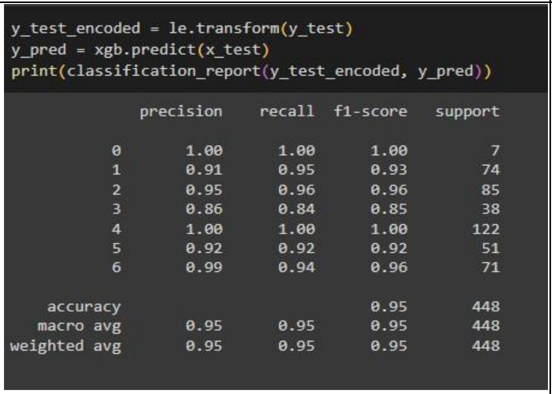
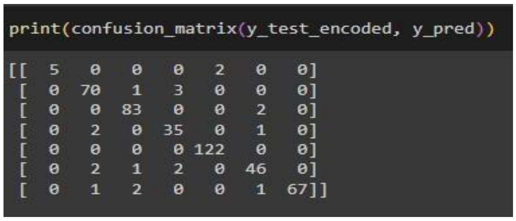
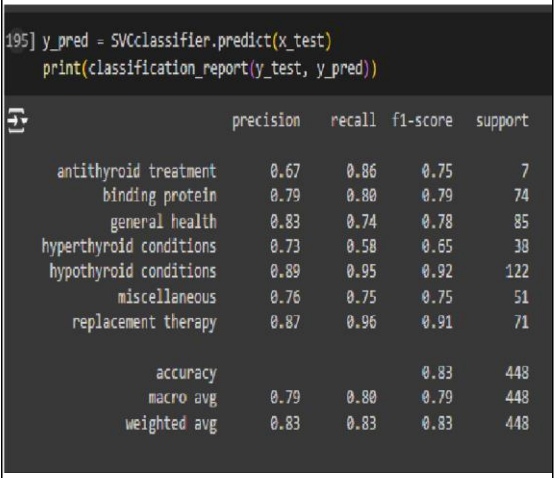
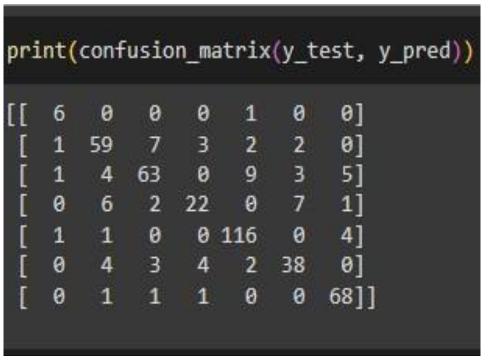
SVC model



```
from sklearn.svm import SVC
SVCclassifier = SVC(kernel='linear', max_iter=251)
SVCclassifier.fit(x_train, y_train)
```

```
SVC
SVC(kernel='linear', max_iter=251)
```

Model Validation and Evaluation Report:

Model	Classification Report	Accuracy	Confusion Matrix
Random Forest Classifier		94.20%	
XGB Classifier		95.54%	
SVC model		86.61%	

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix # Impo

RFclassifier = RandomForestClassifier(max_leaf_nodes=30)
RFclassifier.fit(x_train, y_train)

y_pred = RFclassifier.predict(x_test)

print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred)) # Now you can use confusion_matrix
                         precision    recall  f1-score   support

 antithyroid treatment       0.00      0.00      0.00         7
       binding protein       0.86      0.86      0.86        74
        general health       0.88      0.99      0.93        85
hyperthyroid conditions      0.86      0.82      0.84        38
 hypothyroid conditions      0.95      1.00      0.97       122
         miscellaneous       0.96      0.84      0.90        51
   replacement therapy       0.97      0.94      0.96        71

              accuracy                           0.92       448
             macro avg       0.78      0.78      0.78       448
          weighted avg       0.90      0.92      0.91       448

[[  0   0   1   0   6   0   0]
 [  0  64   6   4   0   0   0]
 [  0   0  84   0   0   0   1]
 [  0   5   0  31   0   1   1]
 [  0   0   0   0 122   0   0]
 [  0   4   2   1   1  43   0]
 [  0   1   2   0   0   1  67]]
```

```
print(confusion_matrix(y_test, y_pred))

[[  3   0   0   0   4   0   0]
 [  0  70   1   3   0   0   0]
 [  0   2  83   0   0   0   0]
 [  0   4   0  31   0   2   1]
 [  0   0   0   0 122   0   0]
 [  0   2   1   1   0  47   0]
 [  0   2   2   0   0   1  66]]
```

```
y_test_encoded = le.transform(y_test)
y_pred = xgb.predict(x_test)
print(classification_report(y_test_encoded, y_pred))

              precision    recall  f1-score   support

           0       1.00      1.00      1.00         7
           1       0.91      0.95      0.93        74
           2       0.95      0.96      0.96        85
           3       0.86      0.84      0.85        38
           4       1.00      1.00      1.00       122
           5       0.92      0.92      0.92        51
           6       0.99      0.94      0.96        71

    accuracy                           0.95       448
   macro avg       0.95      0.95      0.95       448
weighted avg       0.95      0.95      0.95       448
```

```
print(confusion_matrix(y_test_encoded, y_pred))

[[  5   0   0   0   2   0   0]
 [  0  70   1   3   0   0   0]
 [  0   0  83   0   0   2   0]
 [  0   2   0  35   0   1   0]
 [  0   0   0   0 122   0   0]
 [  0   2   1   2   0  46   0]
 [  0   1   2   0   0   1  67]]
```

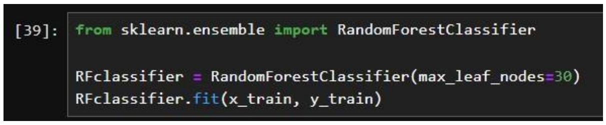
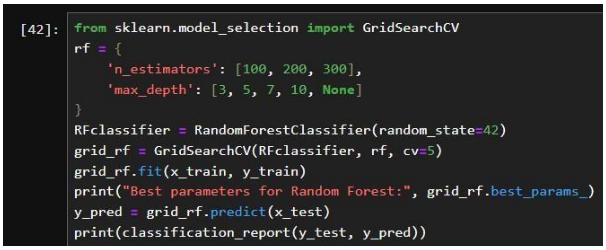
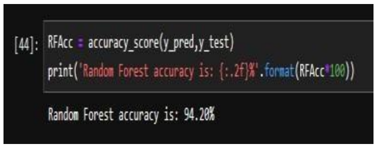
```
195] y_pred = SVCclassifier.predict(x_test)
     print(classification_report(y_test, y_pred))

                         precision    recall  f1-score   support

 antithyroid treatment       0.67      0.86      0.75         7
       binding protein       0.79      0.80      0.79        74
        general health       0.83      0.74      0.78        85
hyperthyroid conditions      0.73      0.58      0.65        38
 hypothyroid conditions      0.89      0.95      0.92       122
         miscellaneous       0.76      0.75      0.75        51
   replacement therapy       0.87      0.96      0.91        71

              accuracy                           0.83       448
             macro avg       0.79      0.80      0.79       448
          weighted avg       0.83      0.83      0.83       448
```

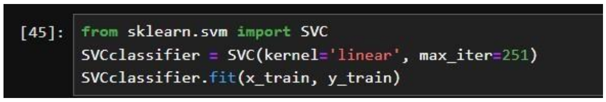
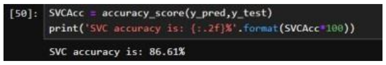
```
print(confusion_matrix(y_test, y_pred))

[[  6   0   0   0   1   0   0]
 [  1  59   7   3   2   2   0]
 [  1   4  63   0   9   3   5]
 [  0   6   2  22   0   7   1]
 [  1   1   0   0 116   0   4]
 [  0   4   3   4   2  38   0]
 [  0   1   1   1   0   0  68]]
```

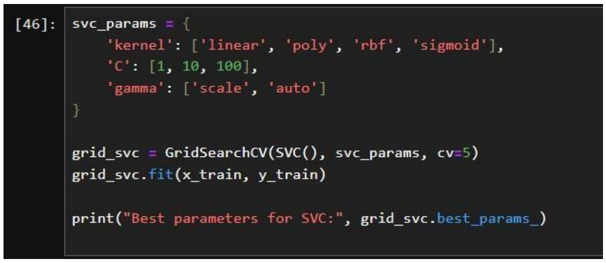
5. Model optimization and Tuning Phase

Hyperparameter Tuning Documentation

Model	Tuned Hyperparameters	Optimal Values

Random Forest
Model
	
	[39]: from sklearn.ensemble import RandomForestClassifier
	
	RFclassifier = RandomForestClassifier(max_leaf_nodes=30)
	RFclassifier.fit(x_train, y_train)
	
	[42]: from sklearn.model_selection import GridSearchCV
	rf = {
	    'n_estimators': [100, 200, 300],
	    'max_depth': [3, 5, 7, 10, None]
	}
	RFclassifier = RandomForestClassifier(random_state=42)
	grid_rf = GridSearchCV(RFclassifier, rf, cv=5)
	grid_rf.fit(x_train, y_train)
	print("Best parameters for Random Forest:", grid_rf.best_params_)
	y_pred = grid_rf.predict(x_test)
	print(classification_report(y_test, y_pred))
		
		[44]: RFAcc = accuracy_score(y_pred,y_test)
		print('Random Forest accuracy is: {:.2f}%'.format(RFAcc*100))
		
		Random Forest accuracy is: 94.20%

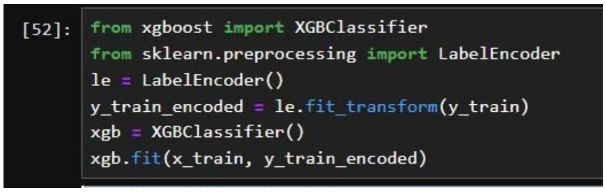
SVC Model	
	[45]: from sklearn.svm import SVC
	SVCclassifier = SVC(kernel='linear', max_iter=251)
	SVCclassifier.fit(x_train, y_train)
		
		[50]: SVCAcc = accuracy_score(y_pred,y_test)
		print('SVC accuracy is: {:.2f}%'.format(SVCAcc*100))
		
		SVC accuracy is: 86.61%



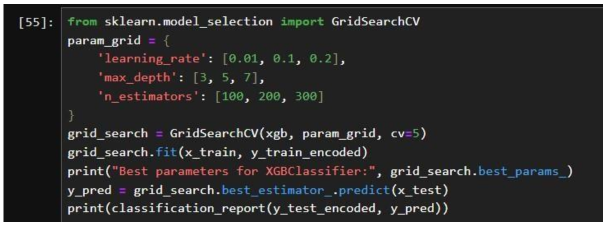
```
[46]: svc_params = {
          'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
          'C': [1, 10, 100],
          'gamma': ['scale', 'auto']
      }

      grid_svc = GridSearchCV(SVC(), svc_params, cv=5)
      grid_svc.fit(x_train, y_train)

      print("Best parameters for SVC:", grid_svc.best_params_)
```

XGB Classifier

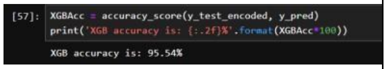
Model



```
[52]: from xgboost import XGBClassifier
      from sklearn.preprocessing import LabelEncoder
      le = LabelEncoder()
      y_train_encoded = le.fit_transform(y_train)
      xgb = XGBClassifier()
      xgb.fit(x_train, y_train_encoded)
```



```
[55]: from sklearn.model_selection import GridSearchCV
      param_grid = {
          'learning_rate': [0.01, 0.1, 0.2],
          'max_depth': [3, 5, 7],
          'n_estimators': [100, 200, 300]
      }
      grid_search = GridSearchCV(xgb, param_grid, cv=5)
      grid_search.fit(x_train, y_train_encoded)
      print("Best parameters for XGBClassifier:", grid_search.best_params_)
      y_pred = grid_search.best_estimator_.predict(x_test)
      print(classification_report(y_test_encoded, y_pred))
```



```
[57]: XGBAcc = accuracy_score(y_test_encoded, y_pred)
      print('XGB accuracy is: {:.2f}%'.format(XGBAcc*100))

      XGB accuracy is: 95.54%
```

Performance Metrics Comparison Report

Model	Baseline Metric	Optimized Metric
Random Forest Classifier	Accuracy: 0.95	Accuracy: 0.9420

XGB Classifier	Accuracy: 0.94	Accuracy: 0.954125
Support Vector Classifier	Accuracy: 0.93	Accuracy: 0.8610714

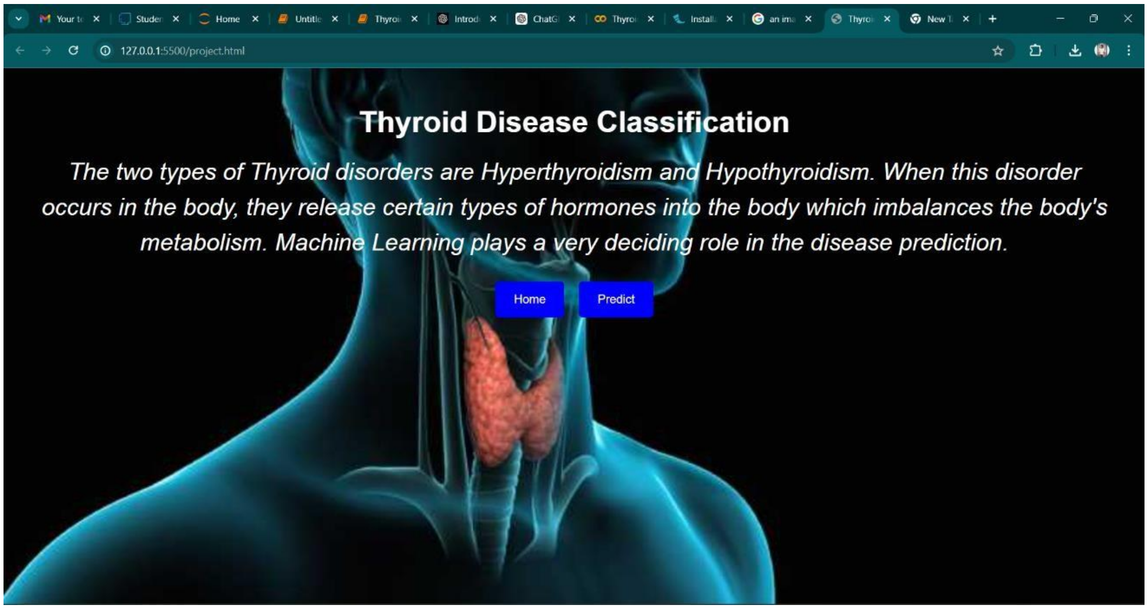
Final Model Selection Justification

Final Model	Reasoning
XGB Classifier	This model is selected for its highest optimized accuracy (0.953125) among the evaluated models, and its robustness in handling imbalanced data.

6. Result

6.1 Output Screenshots

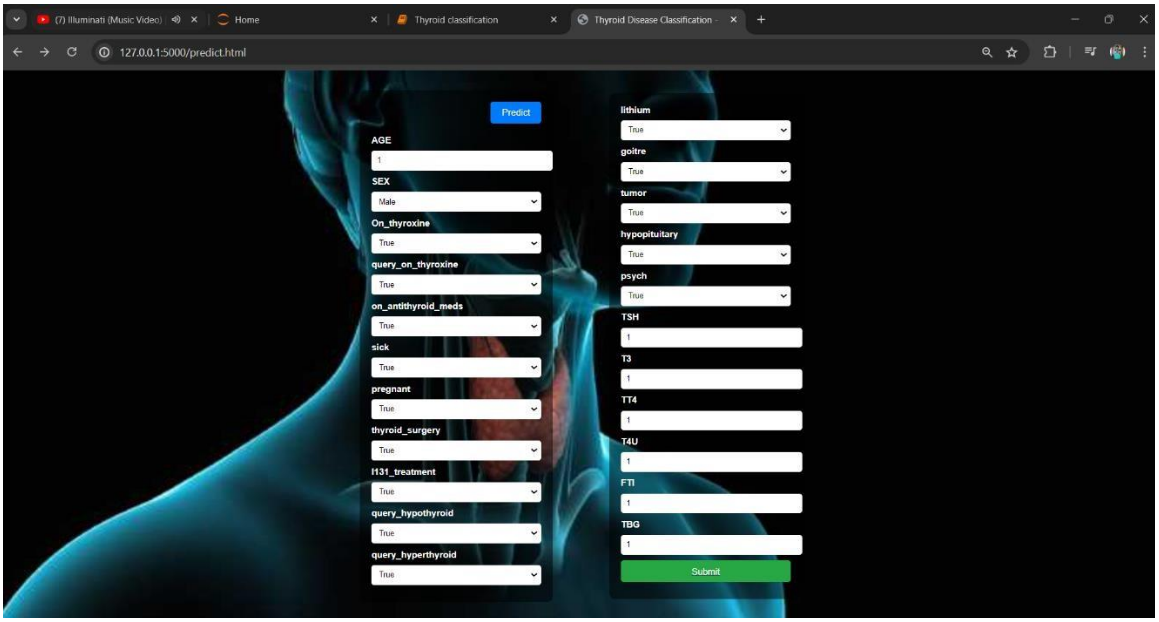
HOME PAGE :-



Thyroid Disease Classification

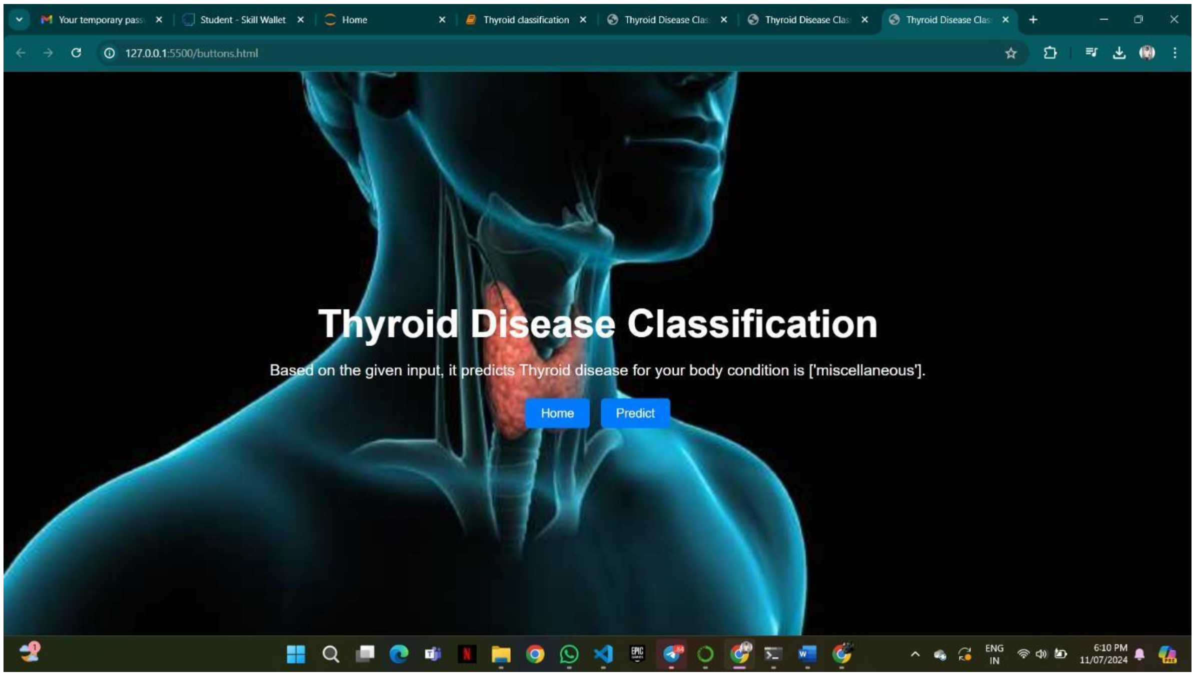
The two types of Thyroid disorders are Hyperthyroidism and Hypothyroidism. When this disorder occurs in the body, they release certain types of hormones into the body which imbalances the body's metabolism. Machine Learning plays a very deciding role in the disease prediction.

PREDICT PAGE:-


(7) Illuminati (Music Video)
Home
Thyroid classification
Thyroid Disease Classification
127.0.0.1:5000/predict.html
Predict
Submit

AGE: 1
SEX: Male
On_thyroxine: True
query_on_thyroxine: True
on_antithyroid_meds: True
sick: True
pregnant: True
thyroid_surgery: True
I131_treatment: True
query_hypothyroid: True
query_hyperthyroid: True
lithium: True
goitre: True
tumor: True
hypopituitary: True
psych: True
TSH: 1
T3: 1
TT4: 1
T4U: 1
FTI: 1
TBG: 1

SUBMIT PAGE:-



Thyroid Disease Classification

Based on the given input, it predicts Thyroid disease for your body condition is ['miscellaneous'].

7. Advantages & Disadvantages

Advantages of Thyroid Classification ML Model:

Early Detection and Diagnosis:

- Speed and Efficiency: Machine learning models can quickly analyze large volumes of patient data to detect thyroid disorders early, leading to timely treatment and better outcomes.
- Accuracy: Advanced algorithms can identify subtle patterns in the data that may be missed by traditional methods, improving diagnostic accuracy.

Personalized Treatment Plans:

- Tailored Recommendations: By analyzing individual patient data, ML models can suggest personalized treatment plans, optimizing therapy for each patient.

Resource Optimization:

- Healthcare Efficiency: Automating the classification process reduces the workload on healthcare professionals, allowing them to focus on patient care.

Enhanced Predictive Analytics:

- Risk Stratification: ML can identify high-risk patients who may need closer monitoring or more aggressive treatment.

Comprehensive Data Utilization:

- Multivariate Analysis: ML can handle and analyze complex, multivariate data, such as hormone levels, patient history, and genetic information, providing a comprehensive assessment.

Improvement in Patient Outcomes:

- Better Prognosis: Accurate classification leads to better prognosis and management of thyroid disorders, improving overall patient outcomes.

Scalability:

- Wider Reach: ML models can be deployed in various healthcare settings, from large hospitals to small clinics, making advanced diagnostic tools accessible to more patients.

These advantages highlight the transformative potential of machine learning in the field of thyroid disorder classification, contributing to better healthcare delivery and patient outcomes.

Disadvantages of Thyroid Classification ML Model:

Data Availability and Quality: Acquiring sufficient and high-quality data for training the ML model can be challenging in the medical domain. Medical data often has issues such as noise, missing values, and class imbalance, which can affect the model's performance and generalizability.

Interpretability: Many advanced ML models, particularly deep learning models, can be complex and act as black boxes, making it difficult to interpret how they arrive at their decisions. In medical applications like thyroid classification, interpretability is crucial for understanding the rationale behind diagnoses.

Handling Uncertainty: Medical diagnoses often involve uncertainty due to varying degrees of symptom severity and the subjective nature of symptoms. ML models typically output deterministic predictions, which may not adequately capture this uncertainty.

Ethical and Legal Considerations: Handling sensitive patient data raises ethical concerns related to privacy, consent, and data security. ML models must comply with regulations such as GDPR or HIPAA to protect patient information.

Bias and Fairness: ML models can inadvertently learn biases present in the training data, leading to disparities in diagnoses across different demographic groups. Ensuring fairness and mitigating biases is critical for equitable healthcare outcomes.

Validation and Clinical Adoption: Validating the performance of the ML model against established clinical standards is essential but challenging. Healthcare providers may be hesitant to adopt ML-based tools without robust evidence of their reliability and accuracy.

Integration into Clinical Workflow: Integrating an ML model into existing clinical workflows requires seamless integration and acceptance by healthcare professionals. Changes in workflow or resistance to adopting new technologies can pose barriers to implementation.

8. Conclusion:-

This study effectively uses patient data to classify thyroid problems using machine learning algorithms. Through testing and selection of the most accurate model, we have developed a user-friendly tool for thyroid problem prediction, which we then implemented in a Flask web application. This method improves accessibility and diagnostic accuracy, but it necessitates constant maintenance and cautious data handling. All things considered, the research helps to better healthcare results by early thyroid disease detection and automated diagnosis

9. Future Scope:-

1. Advanced Models: Investigate more accurate machine learning methods.

2. Data Integration: To enhance forecasts, include additional data sources.

3. Real-Time Updates: Permit the model to be updated often with fresh data.

4. User Interface: Add interactive elements to the Flask app.

10.Appendix:-

Github links:-

Team Member 1:- https://github.com/Rohith26-kvs/Thyroid-classification

Team Member 2:- https://github.com/surya2508/Thyroid-classification-

Team Member 3:- https://github.com/jaswanth-2004-11/Thyroid-Classification

Team Member 4:- https://github.com/abhi14842/THYROID-CLASSIFICATION

Project Demo Link :-