THYROID DISEASE DETECTION

KUNAL AGGARWAL

DATE: 10/02/2023

In this report, I am analyzing the detection of thyroid disease in today's generation using various machine learning algorithms. Nowadays, the thyroid disease is very common among the people, especially among women.

By this report, I am trying to show that the person is thyroid positive or thyroid negative. For this prediction, I am using various machine learning classification algorithms based on the collected data and performing some EDA and feature engineering.

1. Problem Statement:

Our thyroid creates and produces hormones that play a role in many different systems throughout our body. When our thyroid makes either too much or too little of these hormones, it's called a thyroid disease. There are several different types of thyroid disease, including hyperthyroidism, hypothyroidism, thyroiditis and Hashimoto's thyroiditis. Thyroid disease is very common, with an estimated 20 million people in the U.S. having some type of thyroid disorder. A woman is about 5 to 8 times more likely to be diagnosed with a thyroid condition than a man. So how can we detect this disease quickly and efficiently? We can tackle this problem using various machine learning algorithms.

2. Market/Customer/Business Need Assessment:

As this thyroid disease is increasing day by day and the people around the world wants to overcome this problem as quickly and effectively as possible. So to tackle this situation this machine learning model can play a vital role.

It can help to detect the thyroid disease very quickly and with very high accuracy and will also be cheaper than other methods. So it can increase the demand of this proposal.

3. Target Specifications and Characterization:

1. Sensitivity and specificity: The accuracy of the test in detecting thyroid diseases, and the ability of the test to differentiate between different types of thyroid disorders.

2. Test method: The type of test to be used for detecting thyroid diseases such as a blood test, ultrasound, or biopsy.

3. Test accuracy: The level of accuracy required for the test, taking into account the type of thyroid disease and the stage of disease.

4. Test accessibility: The accessibility of the test, including the cost, time required, and availability of the test in different regions.

5. Test reproducibility: The consistency and reliability of the test results over time and in different settings.

The characterization of the target would describe the characteristics of the population tested, including demographic information, risk factors for thyroid disease, and any other relevant information. This information would be used to inform the design of the test and to evaluate the performance of the test in different populations.

4. External search:

The dataset for thyroid disease detection is available on **Kaggle.** The dataset is designed in such a way that is easy to understand and aims on giving main factors which play a healthy role in detecting thyroid disease. These factors include age, gender, hypothyroid factors, hyperthyroid factors, thyroid levels(TT4, T4U, TSH, T3, FTI) and a target variable which is binary class containing the result of thyroid disease (positive, negative).

https://www.kaggle.com/datasets/yasserhessein/thyroid-disease-data-set

https://archive.ics.uci.edu/ml/datasets/thyroid+disease

5. Bench marking alternate products:

There are many products in the market which are used to detect thyroid disease which are also used by doctors in many hospitals around the world. These are good products but expensive and time consuming. My **AI-powered portable thyroid testing device** can solve this issue easily. Some of the benefits of this platform are:

1. Cost: It takes very less cost as compared to the fees taken by the hospitals.

2. Ease of use: It is very easy to use as compared to other thyroid testing devices.

3. Accuracy: As it is AI based product so it gives very high accuracy as compared to the devices used in various hospitals.

4. Time saving: People can test easily at their home without rushing to the hospital which saves lot of time.

5. Availability: It will be easily available on various selling platforms like Amazon, Flipkart, etc.

6. Applicable Patents:

There is one patent named **mobile architecture using cloud for Hashimoto's Thyroiditis disease Classification** which was filed by Suri Jasjit S but got abandoned. It uses cloud architecture and machine learning method to detect the disease.

7. Applicable Regulations:

The regulations surrounding the use of machine learning in thyroid disease detection devices may vary by country. In general, medical devices must be approved by regulatory agencies such as the U.S. Food and Drug Administration (FDA) or the European Union's European Medicines Agency (EMA) before they can be marketed and used for patient care. In India, the regulation of medical devices is governed by the Central Drugs Standard Control Organization (CDSCO) under the Ministry of Health and Family Welfare. The following regulations are applicable in India:

1. Medical Devices Rules, 2017

2. The Indian Medical Devices (Marketing Authorization) Regulations, 2017

3. The Indian Medical Devices (Central Drugs Standard Control Organization) Amendment Rules, 2020

4. The Personal Data Protection Bill, 2019

8. Applicable Constraints:

1. Data Privacy: The collection and processing of sensitive medical data, such as thyroid health information, can raise concerns about data privacy and security. The device must comply with relevant privacy laws and regulations, such as the General Data Protection Regulation (GDPR) to ensure the protection of patient data.

2. Clinical Validation: The accuracy and reliability of the device must be validated through clinical trials and independent studies. This can be a time-consuming and expensive process, and may limit the speed at which the device can be brought to market.

3. Technical Complexity: Machine learning algorithms are complex and require extensive data and computational resources to develop and deploy effectively. The development and deployment of the device may also require specialized expertise in machine learning, data science, and healthcare.

4. Regulation: The device must comply with relevant regulations, such as the FDA's 510(k) clearance process or the European Union's Medical Device Regulation (MDR), to ensure the safety and efficacy of the device.

5. Cost: The cost of developing, manufacturing, and deploying the device, as well as the cost of training healthcare providers to use the device, can be significant and may limit its adoption.

9. Business Model:

1. Device Sales: The device can be sold directly to consumers, healthcare providers, or hospitals. The cost of the device can vary depending on its features and capabilities as well as the target market.

2. Subscription model: The device can be offered as a subscription-based service, where users pay a monthly or annual fee to access the device and receive regular updates and upgrades.

3. Data Monetization: The device can collect and process vast amounts of health data, which can be monetized by selling data insights to healthcare organizations, pharmaceutical companies.

4. Advertising: The device can be integrated with targeted advertising and marketing, where advertisements are displayed on the device or in its associated app.

5. Partnerships: The device can be offered as part of a package deal with healthcare providers, insurance companies, or hospitals, where the device is provided to patients as part of their treatment plan.

10. Concept Generation:

1. Identify the problem: The first step is to identify the problem that the device will address. In this case, the problem is the accurate and timely diagnosis of thyroid diseases.

2. Define the goals and objectives: The next step is to define the goals and objectives of the device. For example, the device should be able to analyze medical data, such as imaging and blood tests, to detect and diagnose thyroid

diseases accurately, quickly, and with high sensitivity and specificity.

3. Determine the feasibility and impact: The feasibility of the solution can be evaluated by assessing the availability and quality of the data, the expertise and resources required, and the technical and regulatory challenges that need to be overcome. The impact of the solution can be evaluated by assessing the potential benefits to patients, healthcare providers, and the healthcare system, such as improved diagnostic accuracy, reduced healthcare costs, and better patient outcomes.

4. Choose the appropriate machine learning algorithms and techniques: The next step is to choose the appropriate machine learning algorithms and techniques that will be used to build the device. This may involve supervised or unsupervised learning, deep learning, natural language processing, or other techniques, depending on the nature of the data and the goals of the device.

5. Define the data requirements: The data requirements for the device should be defined, including the type, quality, and quantity of data that is needed. This may involve collecting and labeling new data, accessing existing datasets, or partnering with healthcare organizations to obtain the necessary data.

6. Design the system architecture: The next step is to design the system architecture, including the software and hardware components that will be required. This may involve choosing a cloud-based or on-premise solution, selecting appropriate programming languages and frameworks, and designing an efficient and scalable system architecture.

7. Develop a prototype: The final step is to develop a prototype of the device, which can be tested and refined in a clinical or research setting. This may involve designing a user interface, integrating with medical imaging and electronic health record systems, and validating the accuracy and performance of the device.

11. Concept Development:

An AI-powered thyroid disease detection device using machine learning is a computer system that uses machine learning algorithms to analyze medical data, such as imaging and blood tests, to help detect and diagnose thyroid diseases. Machine learning algorithms can identify patterns and correlations within large datasets that may be difficult for human clinicians to recognize. These systems can help healthcare providers to quickly and accurately diagnose thyroid diseases, potentially improving patient outcomes and reducing healthcare costs. To develop such a device, a multidisciplinary team of experts is needed, including data scientists, healthcare professionals, data analysts, software developers, and project managers. The development process requires selecting appropriate machine learning algorithms, frameworks, and software, properly labeling and pre-processing data, and following best practices for software development and deployment. It's important to ensure that these devices are validated, regulated, and compliant with appropriate ethical and legal standards.

12. Final Product Prototype:

1. User interface: The device would have a user-friendly interface for healthcare professionals to input patient data and view diagnostic results.

2. Data collection and preprocessing: The device would collect medical data such as medical images and blood test results from electronic health records, and pre-process the data by normalizing, standardizing, and cleaning the data.

3. Machine learning algorithms: The device would use machine learning algorithms to analyze the pre-processed data, identify patterns and correlations, and diagnose thyroid diseases with high accuracy.

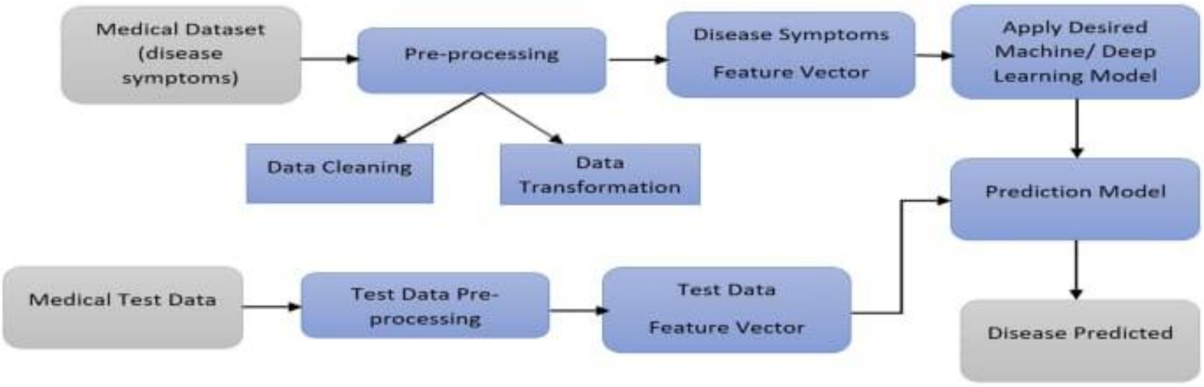
4. Diagnosis report: The device would generate a diagnosis report that summarizes the diagnostic results and provides information about the detected thyroid disease.

5. Performance metrics: The device would include performance metrics such as sensitivity, specificity, and accuracy to measure the performance of the machine learning algorithms.

6. Regulatory compliance: The device would comply with regulatory requirements such as FDA approval, data privacy and security regulations, and other ethical and legal standards.

7. Scalability: The device would be designed to be scalable, allowing it to handle large amounts of data and be used in a variety of healthcare settings.

8. Cloud-based or on - premise solution: The device could be designed as a cloud-based or on - premise solution, depending on the needs and preferences of healthcare organizations.



13. Product Details:

1. Working:

1. Data Collection: The first step is to collect a large and diverse dataset of thyroid-related information. This dataset may include medical records, blood test results, imaging data (such as ultrasound or CT scans), and other relevant patient information.

2. Data Preprocessing: The collected data is then preprocessed to ensure that it is of high quality, is properly labeled, and is ready to be used by machine learning algorithms. This may involve removing any irrelevant or duplicate information, filling in missing data, and normalizing the data to ensure consistency.

3. Feature Extraction: The next step is to extract relevant features from the preprocessed data. This may involve using various statistical techniques to identify the most important features that can help in accurately identifying thyroid-related issues.

4. Model Selection: Once the features are extracted, a suitable machine learning algorithm is selected for the task. Commonly used algorithms include decision trees, logistic regression, random forests, and deep learning algorithms like neural networks.

5. Model Training: The machine learning model is then trained on the preprocessed and feature extracted data. This involves feeding the algorithm with labeled data and allowing it to learn the patterns and relationships between the input features and the output labels.

6. Model Evaluation: Once the model is trained, it is evaluated on a separate set of data that was not used during the training process. This is to ensure that the model is accurate and robust, and is not overfitting or underfitting the data.

7. Deployment: Once the model is deemed accurate and reliable, it can be deployed in a real-world setting. This may involve integrating the model into a software application or medical device that can be used by healthcare professionals to assist in diagnosing thyroid-related issues.

2. Data Sources:

1. Electronic Health Records (EHRs): EHRs are a rich source of patient data, including medical history, laboratory test results, and imaging data. This data can be used to train machine learning algorithms to accurately diagnose thyroid diseases.

2. Blood Test Results: Blood tests can provide valuable information about thyroid function, including levels of thyroid hormones and antibodies. This data can be used to identify patterns that may indicate the presence of thyroid disease.

3. Imaging Data: Imaging data, such as ultrasound or CT scans, can provide detailed information about the structure and function of the thyroid gland. Machine learning algorithms can be trained to identify patterns in this data that may indicate the presence of thyroid disease.

4. Genetic Data: Genetic data can provide insights into a patient's risk for developing thyroid disease. Machine learning algorithms can be trained to identify genetic markers that are associated with an increased risk of thyroid disease.

5. Patient Surveys: Patient surveys can provide information about symptoms, lifestyle factors, and other factors that may be associated with thyroid disease. This data can be used to identify patterns that may indicate the presence of thyroid disease.

6. Research Studies: Research studies may provide valuable data on the prevalence and risk factors of thyroid

disease. This data can be used to train machine learning algorithms to accurately identify thyroid disease.

3. Algorithms, frameworks and software needed:

1. Algorithms: Supervised learning algorithms such as logistic regression, support vector machines (SVMs), and neural networks (e.g., convolutional neural networks) to build predictive models for disease detection. Unsupervised learning algorithms such as clustering and dimensionality reduction techniques like principal component analysis (PCA) to identify patterns in large datasets and gain insights into disease subtypes.

2. Frameworks: Python-based machine learning frameworks such as TensorFlow, PyTorch, and scikit-learn, which provide pre-built libraries for data pre-processing, model development, and evaluation. Deep learning- specific frameworks like Keras or OpenCV for image recognition and object detection.

3. Software: Cloud platforms such as amazon web services (AWS), google cloud, and Microsoft azure can provide the computational resources required to train and run machine learning models. IDEs such as pycharm and jupyter notebook can be used to write and execute the code and manage the development process. Git and GitHub can be used to manage the source code and collaborate with other developers. Containerization technologies such as docker and

kubernetes can be used to deploy the AI-powered thyroid disease detection device to production.

4. Visualization tools: Visualization libraries like matplotlib or seaborn to help with data exploration and model interpretation.

4. Team Required to Develop:

1. Data scientists and machine learning engineers with expertise in developing predictive models and working with large datasets.

2. Medical doctors and healthcare professionals with domain knowledge of thyroid disease, including clinical presentation, diagnosis, and treatment.

3. Data analysts and statisticians with expertise in data pre-processing, visualization, and statistical analysis.

4. Software developers with expertise in software engineering, cloud computing, and deployment.

5. Project managers with experience in managing cross-functional teams, defining project goals and timelines, and overseeing the project's progress.

6. Quality assurance specialists with expertise in testing, validation, and regulatory compliance.

4. What does it Cost?

The cost of developing an AI-powered thyroid disease detection device using machine learning can vary widely depending on several factors such as the complexity of the technology, the amount of data required, the regulatory requirements, and the expertise and resources of the development team.

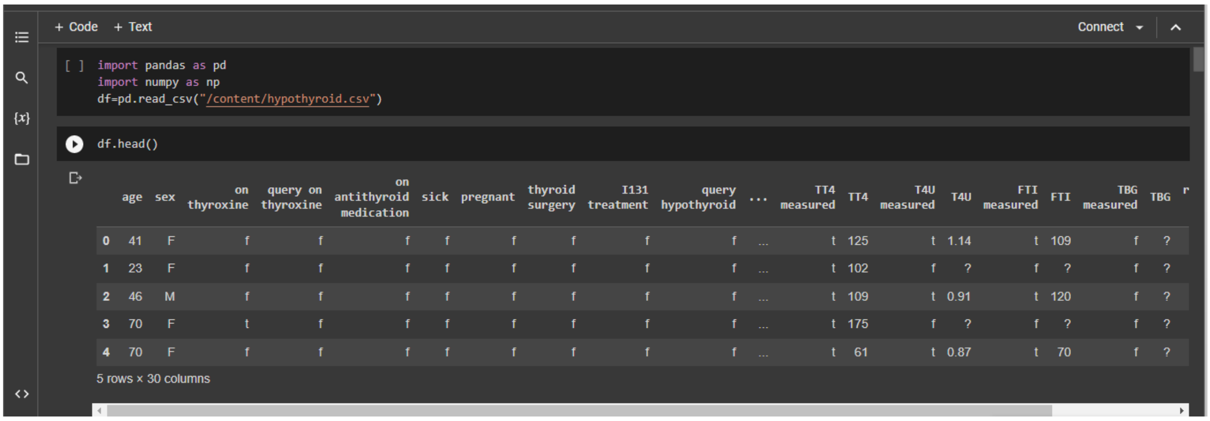
Here are some of the main cost drivers that can contribute to the overall cost of developing an AI-powered thyroid disease detection device:

1. Data collection and preprocessing: The cost of acquiring and preprocessing large amounts of medical data such as medical images and blood tests can be significant.
2. Machine learning algorithms: The cost of developing and testing machine learning algorithms, including the selection and optimization of the algorithms, can be high.
3. Hardware and software infrastructure: The cost of the hardware and software infrastructure required to build and run the machine learning models, including cloud services, servers, and software tools, can also be significant.
4. Regulatory compliance: The cost of obtaining regulatory approval, complying with data privacy and security regulations, and other ethical and legal standards can be high.
5. Expertise and resources: The cost of hiring and retaining a team of experts in data science, healthcare,

software development, and project management can be high.

14. Code Implementation:

1. Loading and reading the Dataset:



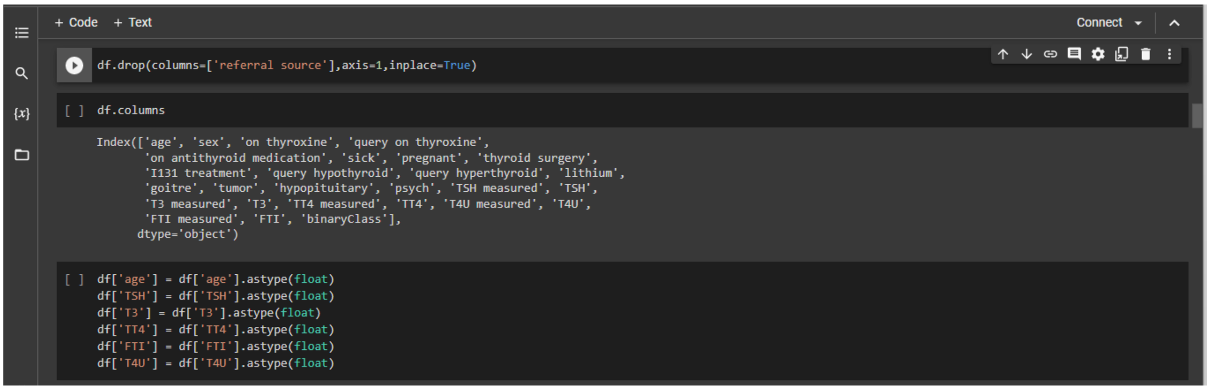
```
[ ] import pandas as pd
    import numpy as np
    df=pd.read_csv("/content/hypothyroid.csv")
```

```
df.head()
```

	age	sex	on thyroxine	query on thyroxine	on antithyroid medication	sick	pregnant	thyroid surgery	I131 treatment	query hypothyroid	...	TT4 measured	TT4	T4U measured	T4U	FTI measured	FTI	TBG measured	TBG	r
0	41	F	f	f	f	f	f	f	f	f	...	t	125	t	1.14	t	109	f	?	
1	23	F	f	f	f	f	f	f	f	f	...	t	102	f	?	f	?	f	?	
2	46	M	f	f	f	f	f	f	f	f	...	t	109	t	0.91	t	120	f	?	
3	70	F	t	f	f	f	f	f	f	f	...	t	175	f	?	f	?	f	?	
4	70	F	f	f	f	f	f	f	f	f	...	t	61	t	0.87	t	70	f	?	

5 rows × 30 columns

2. Exploratory Data Analysis (EDA):

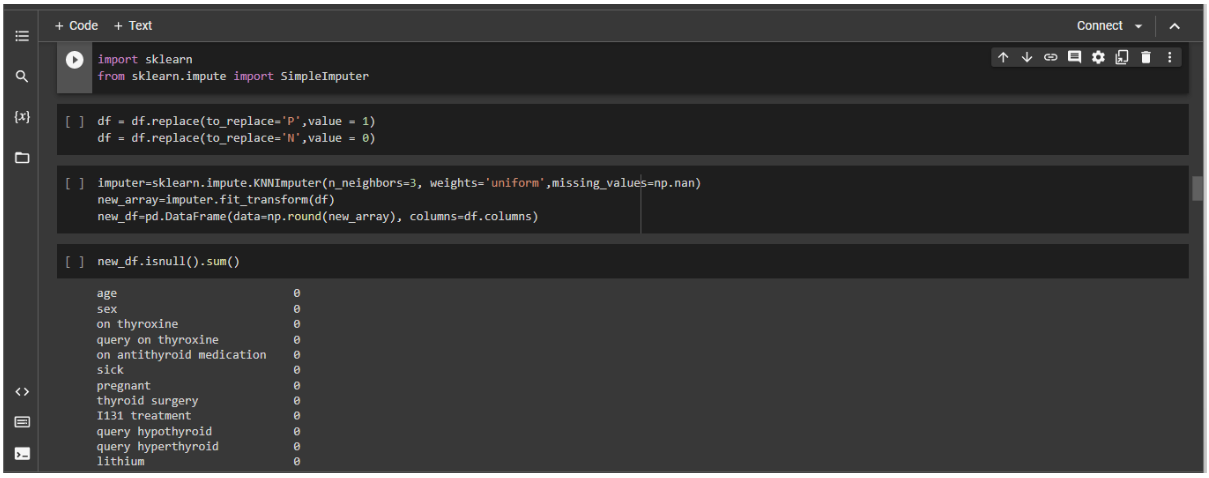


```
df.drop(columns=['referral source'],axis=1,inplace=True)
```

```
[ ] df.columns
```

```
Index(['age', 'sex', 'on thyroxine', 'query on thyroxine',
       'on antithyroid medication', 'sick', 'pregnant', 'thyroid surgery',
       'I131 treatment', 'query hypothyroid', 'query hyperthyroid', 'lithium',
       'goitre', 'tumor', 'hypopituitary', 'psych', 'TSH measured', 'TSH',
       'T3 measured', 'T3', 'TT4 measured', 'TT4', 'T4U measured', 'T4U',
       'FTI measured', 'FTI', 'binaryClass'],
      dtype='object')
```

```
[ ] df['age'] = df['age'].astype(float)
    df['TSH'] = df['TSH'].astype(float)
    df['T3'] = df['T3'].astype(float)
    df['TT4'] = df['TT4'].astype(float)
    df['FTI'] = df['FTI'].astype(float)
    df['T4U'] = df['T4U'].astype(float)
```



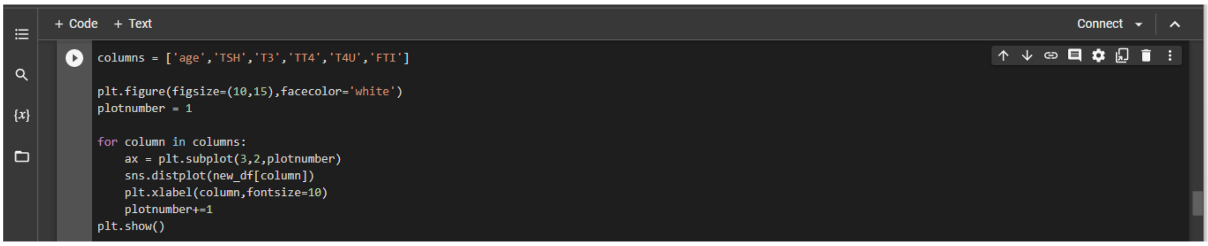
```
import sklearn
from sklearn.impute import SimpleImputer
```

```
[ ] df = df.replace(to_replace='P',value = 1)
    df = df.replace(to_replace='N',value = 0)
```

```
[ ] imputer=sklearn.impute.KNNImputer(n_neighbors=3, weights='uniform',missing_values=np.nan)
    new_array=imputer.fit_transform(df)
    new_df=pd.DataFrame(data=np.round(new_array), columns=df.columns)
```

```
[ ] new_df.isnull().sum()
```

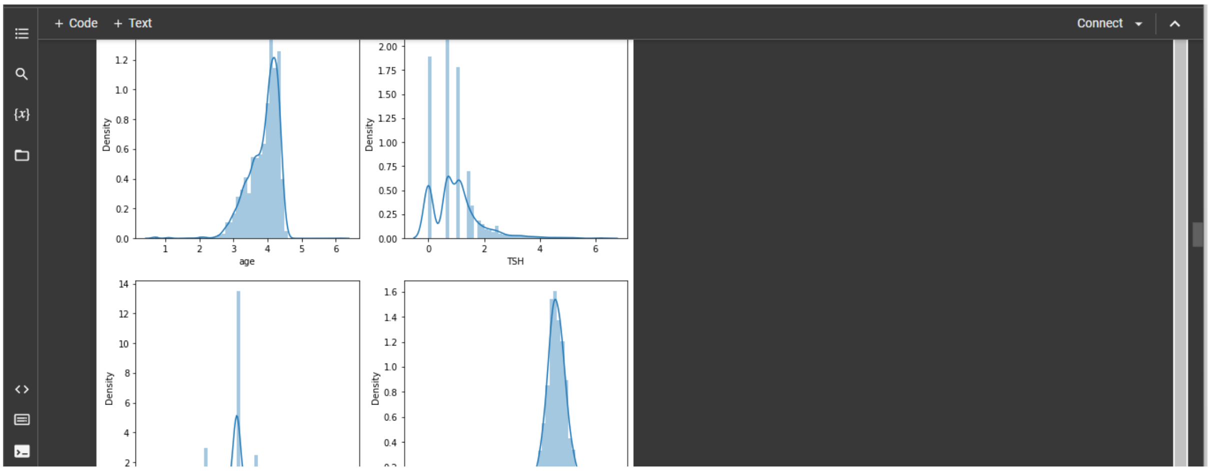
```
age                          0
sex                          0
on thyroxine                 0
query on thyroxine           0
on antithyroid medication    0
sick                         0
pregnant                     0
thyroid surgery              0
I131 treatment               0
query hypothyroid            0
query hyperthyroid           0
lithium                      0
```

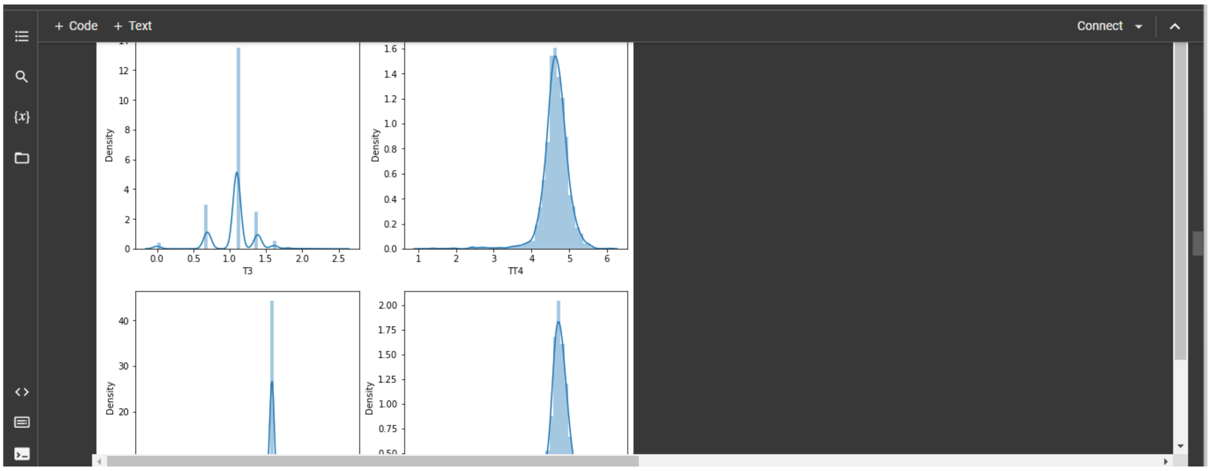


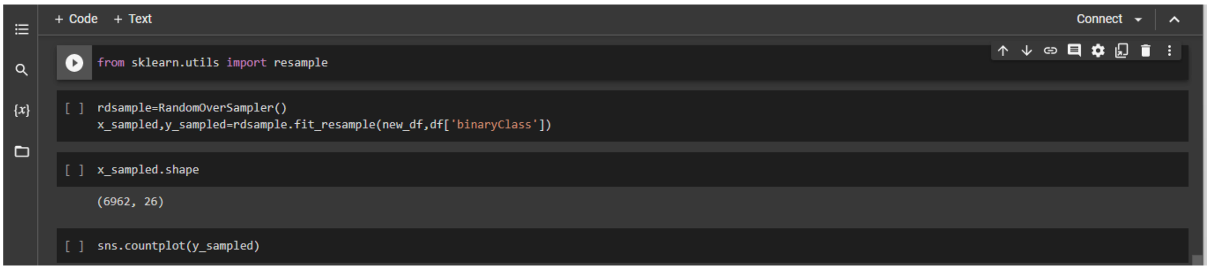
```
columns = ['age','TSH','T3','TT4','T4U','FTI']

plt.figure(figsize=(10,15),facecolor='white')
plotnumber = 1

for column in columns:
    ax = plt.subplot(3,2,plotnumber)
    sns.distplot(new_df[column])
    plt.xlabel(column,fontsize=10)
    plotnumber+=1
plt.show()
```







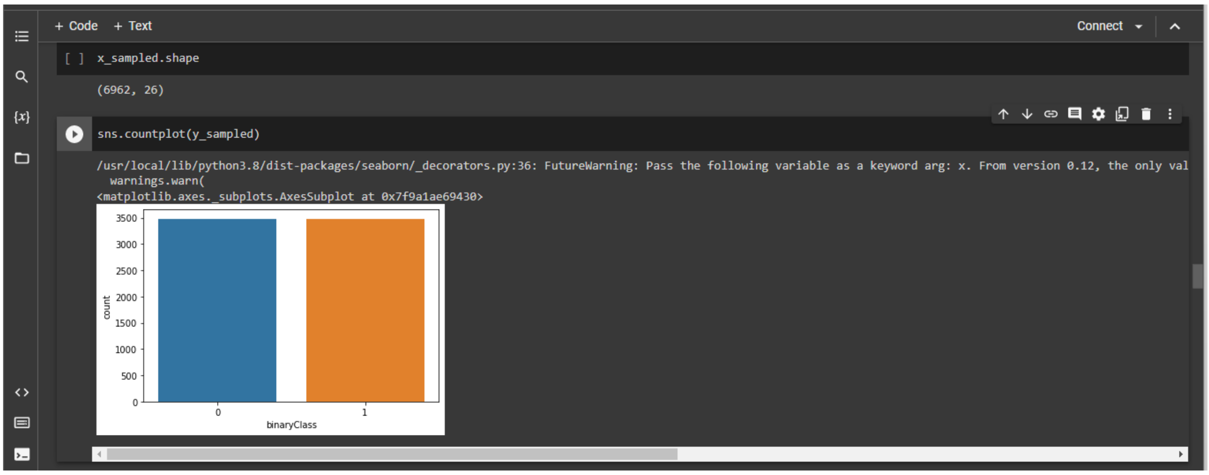
```
from sklearn.utils import resample
```

```
[ ] rdsample=RandomOverSampler()
    x_sampled,y_sampled=rdsample.fit_resample(new_df,df['binaryClass'])
```

```
[ ] x_sampled.shape

    (6962, 26)
```

```
[ ] sns.countplot(y_sampled)
```



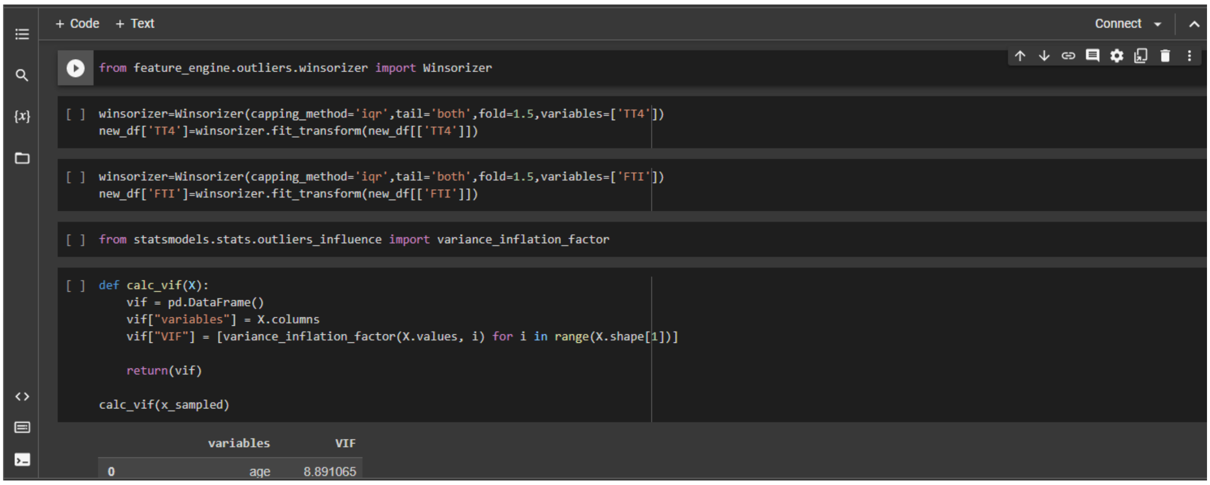
```
[ ] x_sampled.shape

    (6962, 26)
```

```
sns.countplot(y_sampled)

/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only val
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7f9a1ae69430>
```

3. Feature Engineering:



```
from feature_engine.outliers.winsorizer import Winsorizer
```

```
winsorizer=Winsorizer(capping_method='iqr',tail='both',fold=1.5,variables=['TT4'])
new_df['TT4']=winsorizer.fit_transform(new_df[['TT4']])
```

```
winsorizer=Winsorizer(capping_method='iqr',tail='both',fold=1.5,variables=['FTI'])
new_df['FTI']=winsorizer.fit_transform(new_df[['FTI']])
```

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

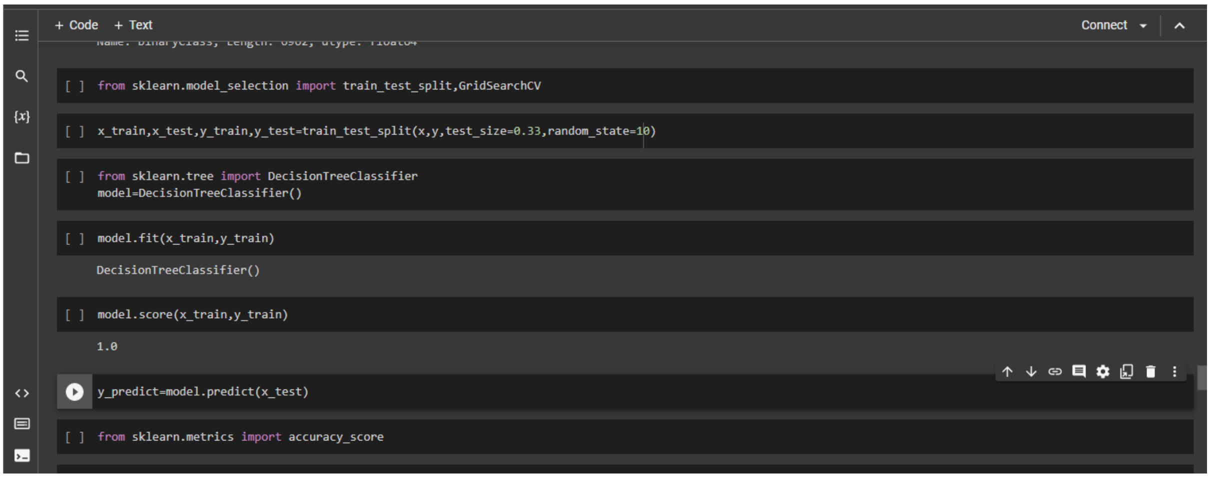
```
def calc_vif(X):
    vif = pd.DataFrame()
    vif["variables"] = X.columns
    vif["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.shape[1])]

    return(vif)

calc_vif(x_sampled)
```

	variables	VIF
0	age	8.891065

4. Splitting the dataset in x, y variable:



```
Name: binaryClass, Length: 6902, dtype: float64
```

```
from sklearn.model_selection import train_test_split,GridSearchCV
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=10)
```

```
from sklearn.tree import DecisionTreeClassifier
model=DecisionTreeClassifier()
```

```
model.fit(x_train,y_train)
```

```
DecisionTreeClassifier()
```

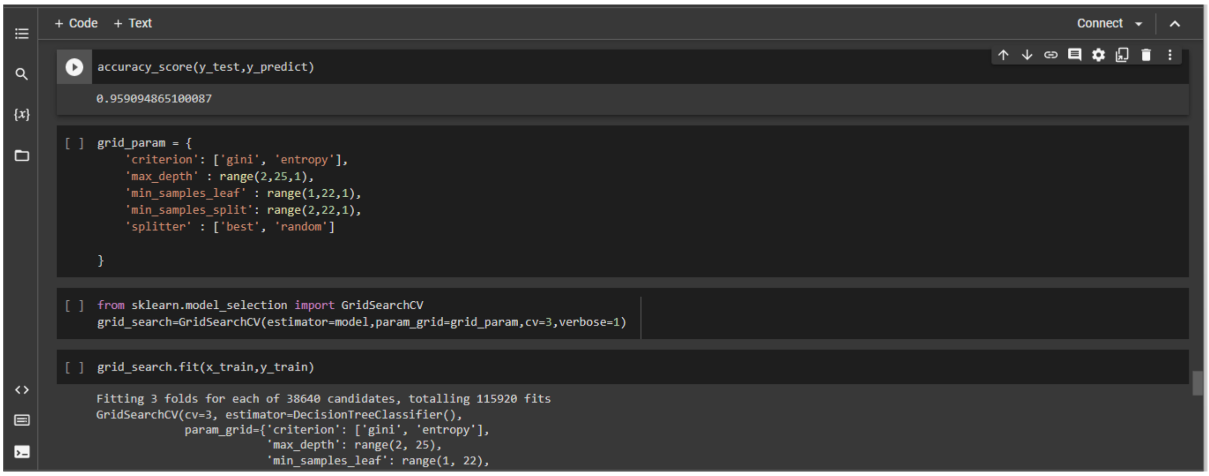
```
model.score(x_train,y_train)
```

```
1.0
```

```
y_predict=model.predict(x_test)
```

```
from sklearn.metrics import accuracy_score
```

5. Preparing the model with hyperparameter tuning:



```
accuracy_score(y_test,y_predict)
```

```
0.959094865100087
```

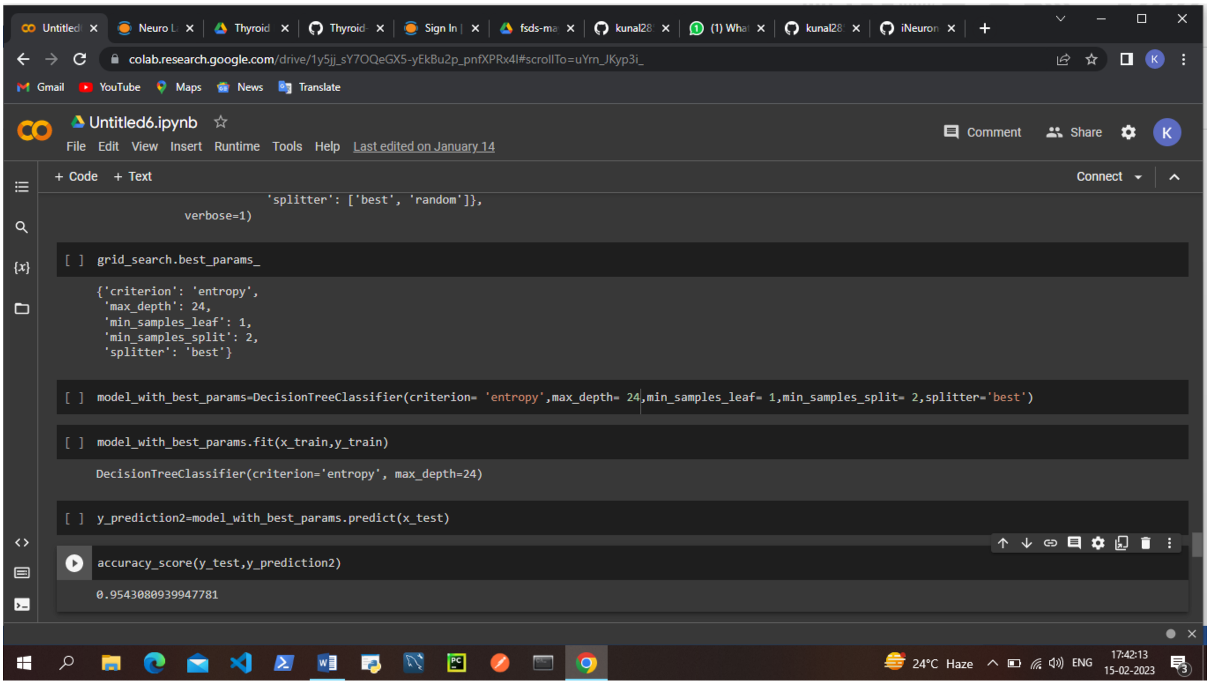
```
grid_param = {
    'criterion': ['gini', 'entropy'],
    'max_depth' : range(2,25,1),
    'min_samples_leaf' : range(1,22,1),
    'min_samples_split': range(2,22,1),
    'splitter' : ['best', 'random']

}
```

```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=model,param_grid=grid_param,cv=3,verbose=1)
```

```
grid_search.fit(x_train,y_train)
```

```
Fitting 3 folds for each of 38640 candidates, totalling 115920 fits
GridSearchCV(cv=3, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': range(2, 25),
                         'min_samples_leaf': range(1, 22),
```



Untitled6.ipynb
File Edit View Insert Runtime Tools Help Last edited on January 14
Comment Share
+ Code + Text
Connect
colab.research.google.com/drive/1y5jj_sY7OQeGX5-yEkBu2p_pnfXPRx4l#scrollTo=uYrn_JKyp3i_
24°C Haze ENG 17:42:13 15-02-2023

```
                     'splitter': ['best', 'random']},
             verbose=1)
```

```
[ ] grid_search.best_params_
```

```
{'criterion': 'entropy',
 'max_depth': 24,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'splitter': 'best'}
```

```
[ ] model_with_best_params=DecisionTreeClassifier(criterion= 'entropy',max_depth= 24,min_samples_leaf= 1,min_samples_split= 2,splitter='best')
```

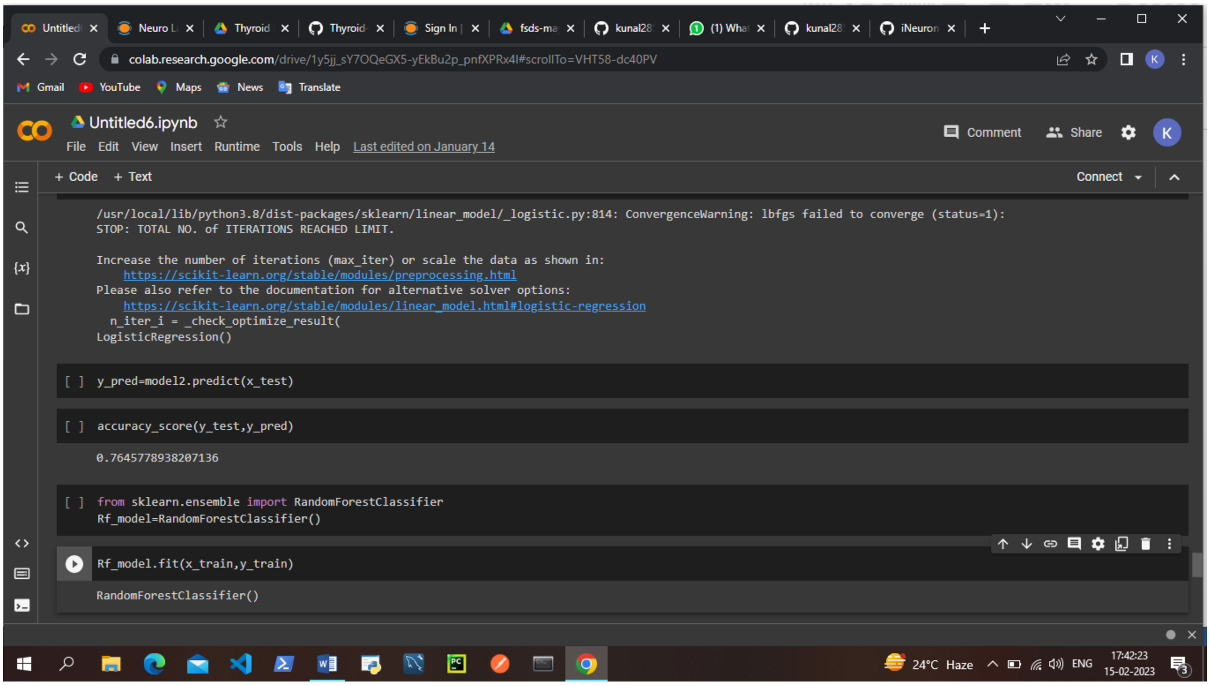
```
[ ] model_with_best_params.fit(x_train,y_train)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=24)
```

```
[ ] y_prediction2=model_with_best_params.predict(x_test)
```

```
accuracy_score(y_test,y_prediction2)
```

```
0.9543080939947781
```



Untitled6.ipynb
File Edit View Insert Runtime Tools Help Last edited on January 14
Comment Share
+ Code + Text
Connect
colab.research.google.com/drive/1y5jj_sY7OQeGX5-yEkBu2p_pnfXPRx4l#scrollTo=VHT58-dc40PV
24°C Haze ENG 17:42:23 15-02-2023

```
/usr/local/lib/python3.8/dist-packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
LogisticRegression()
```

```
[ ] y_pred=model2.predict(x_test)
```

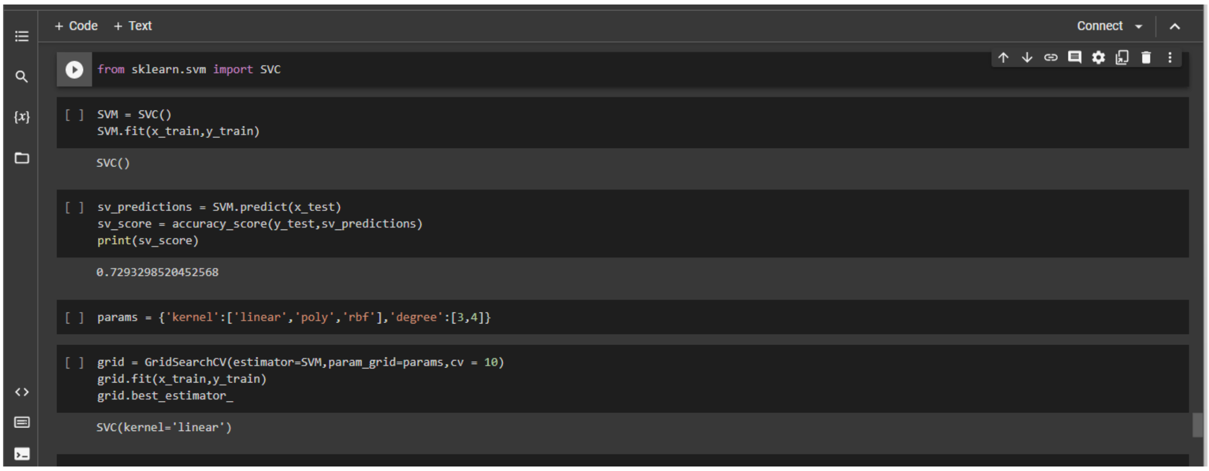
```
[ ] accuracy_score(y_test,y_pred)
```

```
0.7645778938207136
```

```
[ ] from sklearn.ensemble import RandomForestClassifier
    Rf_model=RandomForestClassifier()
```

```
Rf_model.fit(x_train,y_train)
```

```
RandomForestClassifier()
```



```
from sklearn.svm import SVC
```

```
SVM = SVC()
SVM.fit(x_train,y_train)
```

```
SVC()
```

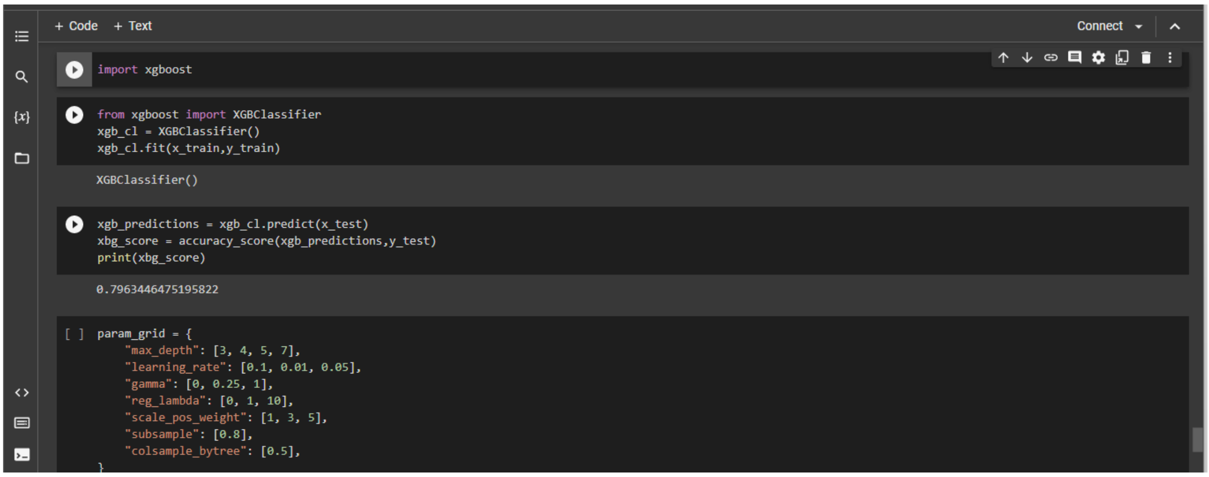
```
sv_predictions = SVM.predict(x_test)
sv_score = accuracy_score(y_test,sv_predictions)
print(sv_score)
```

```
0.7293298520452568
```

```
params = {'kernel':['linear','poly','rbf'],'degree':[3,4]}
```

```
grid = GridSearchCV(estimator=SVM,param_grid=params,cv = 10)
grid.fit(x_train,y_train)
grid.best_estimator_
```

```
SVC(kernel='linear')
```



```
import xgboost
```

```
from xgboost import XGBClassifier
xgb_cl = XGBClassifier()
xgb_cl.fit(x_train,y_train)
```

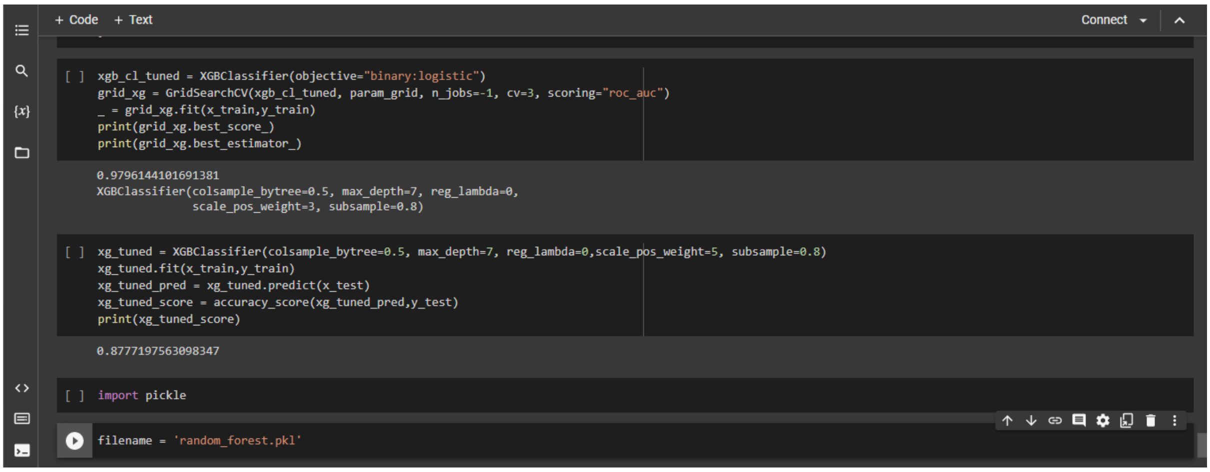
```
XGBClassifier()
```

```
xgb_predictions = xgb_cl.predict(x_test)
xbg_score = accuracy_score(xgb_predictions,y_test)
print(xbg_score)
```

```
0.7963446475195822
```

```
param_grid = {
    "max_depth": [3, 4, 5, 7],
    "learning_rate": [0.1, 0.01, 0.05],
    "gamma": [0, 0.25, 1],
    "reg_lambda": [0, 1, 10],
    "scale_pos_weight": [1, 3, 5],
    "subsample": [0.8],
    "colsample_bytree": [0.5],
}
```

6. Pickeling the final model:



```
xgb_cl_tuned = XGBClassifier(objective="binary:logistic")
grid_xg = GridSearchCV(xgb_cl_tuned, param_grid, n_jobs=-1, cv=3, scoring="roc_auc")
_ = grid_xg.fit(x_train,y_train)
print(grid_xg.best_score_)
print(grid_xg.best_estimator_)
```

```
0.9796144101691381
XGBClassifier(colsample_bytree=0.5, max_depth=7, reg_lambda=0,
              scale_pos_weight=3, subsample=0.8)
```

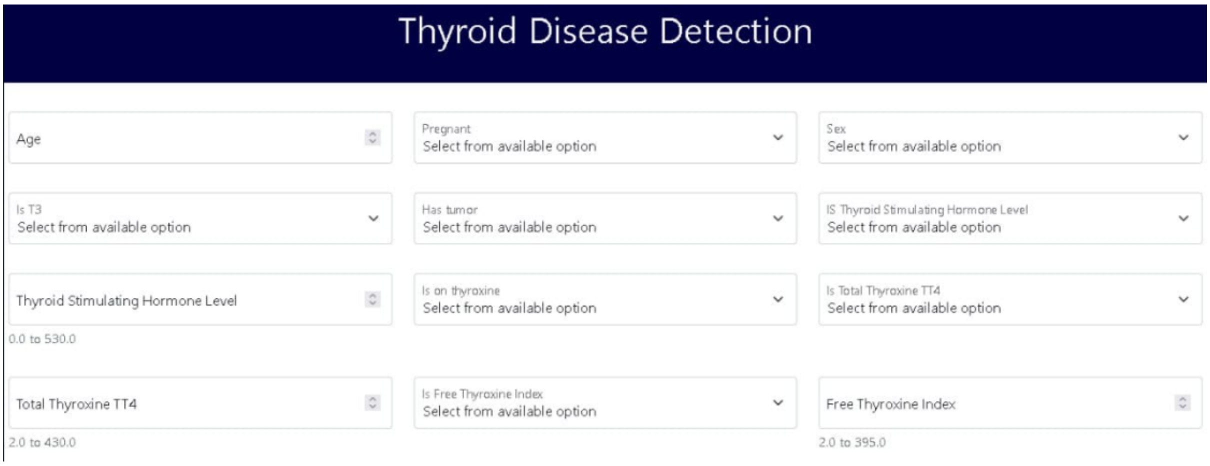
```
xg_tuned = XGBClassifier(colsample_bytree=0.5, max_depth=7, reg_lambda=0,scale_pos_weight=5, subsample=0.8)
xg_tuned.fit(x_train,y_train)
xg_tuned_pred = xg_tuned.predict(x_test)
xg_tuned_score = accuracy_score(xg_tuned_pred,y_test)
print(xg_tuned_score)
```

```
0.8777197563098347
```

```
import pickle
```

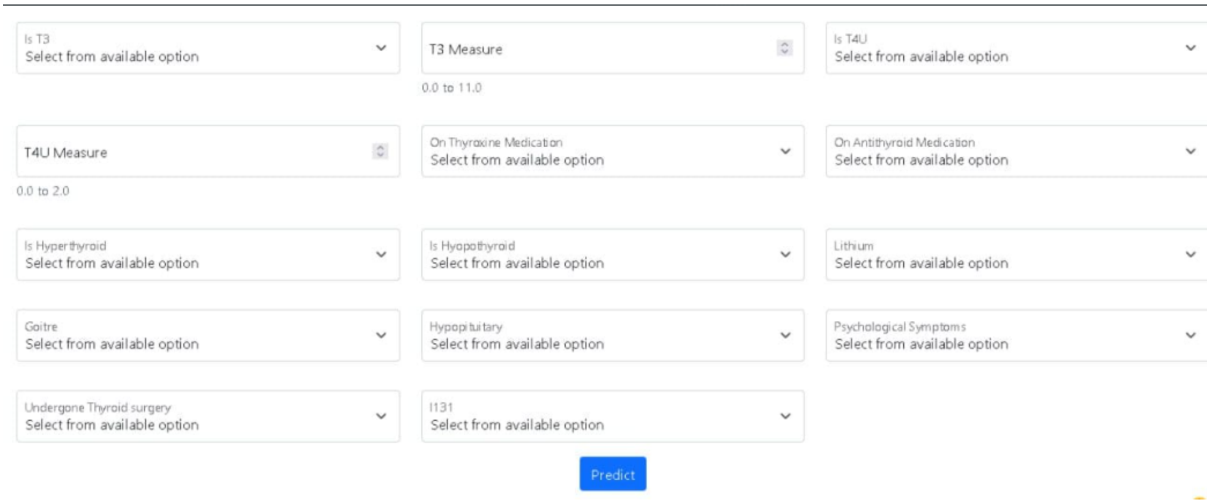
```
filename = 'random_forest.pkl'
```

7. API Page:

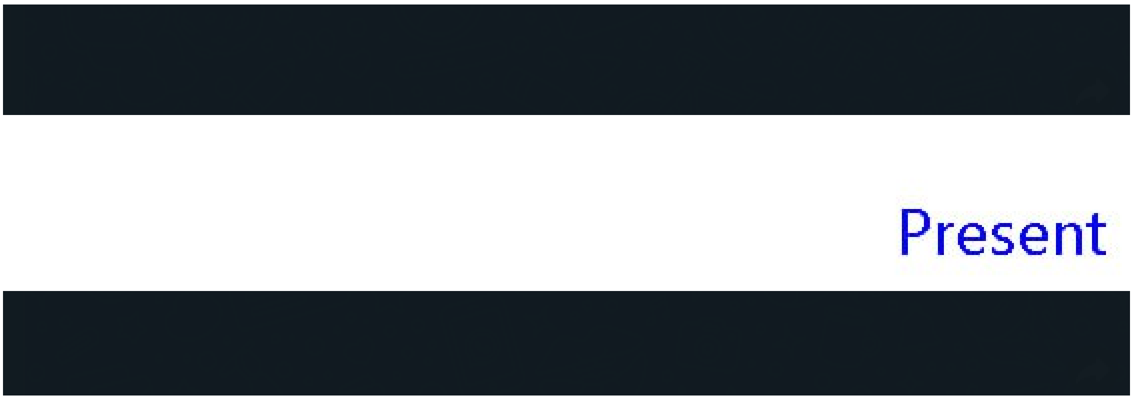


Thyroid Disease Detection

Age	Pregnant Select from available option	Sex Select from available option
Is T3 Select from available option	Has tumor Select from available option	IS Thyroid Stimulating Hormone Level Select from available option
Thyroid Stimulating Hormone Level 0.0 to 530.0	Is on thyroxine Select from available option	Is Total Thyroxine TT4 Select from available option
Total Thyroxine TT4 2.0 to 430.0	Is Free Thyroxine Index Select from available option	Free Thyroxine Index 2.0 to 395.0



Is T3 Select from available option	T3 Measure 0.0 to 11.0	Is T4U Select from available option
T4U Measure 0.0 to 2.0	On Thyroxine Medication Select from available option	On Antithyroid Medication Select from available option
Is Hyperthyroid Select from available option	Is Hypopthyroid Select from available option	Lithium Select from available option
Goitre Select from available option	Hypopituitary Select from available option	Psychological Symptoms Select from available option
Undergone Thyroid surgery Select from available option	I131 Select from available option	



Present

15. Conclusion:

An AI-powered thyroid disease detection device that uses machine learning can offer numerous benefits in improving the accuracy and efficiency of thyroid disease diagnosis. By analyzing patient data and medical images, machine learning algorithms can identify patterns and make predictions with a high degree of accuracy, potentially leading to earlier and more accurate diagnoses, and better patient outcomes.

One potential application of such a device is to screen individuals for thyroid disease, which could help identify patients who are at risk and require further evaluation. Additionally, an AI-powered thyroid disease detection device can assist healthcare providers in making more informed decisions about patient care, leading to more personalized and effective treatment plans.

Overall, an AI-powered thyroid disease detection device has the potential to significantly improve the speed, accuracy, and efficiency of thyroid disease diagnosis, ultimately leading to better patient outcomes and improved healthcare delivery. However, as with any medical technology, it is important to thoroughly validate the accuracy and effectiveness of such a device before integrating it into clinical practice.