Лабораторная работа

Аттестационное задание

Построение классификатора и оценка эффективности его работы

Выполнила: Короткова Инга Сергеевна

2020 год

Цель работы – закрепить навыки проектирования интеллектуальных моделей с помощью библиотеки Scikit-Learn для задач построения классифицирующих систем и оценки эффективности их работы.

Задачи:

Выбрать набор данных, содержащий не менее 500 объектов и не менее 10 атрибутов, отмеченный как набор, для задачи классификации с репозитария https://archive.ics.uci.edu/ml/datasets.php Загрузить выбранный набор данных. Произвести исследовательский анализ данных:

- получить объём исследуемых данных;
- получить число атрибутов и их типы данных;
- посмотреть распределение числа примеров классов
- если необходимо, выполнить преобразование категориальных атрибутов;
- если необходимо, заполнить пропущенные значения в выборке.

Разделить набор данных на обучающую и тестовую выборку и объяснить это разбиение. Обучить модели классификации на основе алгоритмов трёх алгоритмов (по выбору слушателя курса), отметить почему были выбраны эти алгоритмы Оценить эффективность моделей на тестовой выборке с помощью матрицы неточностей, критериев полноты Recall и точности Precision.

Постановка задачи и описание данных:

Описание переменных и датасет взяты с сайта: http://archive.ics.uci.edu/ml/datasets/thyroid+disease

Этот набор данных содержит информацию о заболеваниях связанных с функцией щитовидной железы.

Данные собраны Garvan Institue в Австралии.

Объекты разделены на несколько классов, всего 29 атрибутов (22 категориальных и 7 численных).

Сами данные описаны следующим образом:

- age: continuous.
- sex: M, F.
- on thyroxine: f, t.
- query on thyroxine: f, t.
- on antithyroid medication: f, t.
- sick: f, t.
- pregnant: f, t.
- thyroid surgery: f, t.
- I131 treatment: f, t.
- query hypothyroid: f, t.
- query hyperthyroid: f, t.
- lithium: f, t.
- goitre: f, t.
- tumor: f, t.
- hypopituitary: f, t.
- psych: f, t.
- TSH measured: f, t.
- TSH: continuous.
- T3 measured: f, t.
- T3: continuous.
- TT4 measured: f, t.
- TT4: continuous.
- T4U measured: f, t.
- T4U: continuous.
- FTI measured: f, t.
- FTI: continuous.
- TBG measured: f, t.
- TBG: continuous.
- referral_source WEST, STMW, SVHC, SVI, SVHD, other.
- class

Целевая переменная - Class.

Импорт необходимых библиотек

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
sns.set(font_scale=1.3)

from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import QuantileTransformer
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer

from sklearn.model_selection import train_test_split

from sklearn.multiclass import OneVsRestClassifier

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import GradientBoostingClassifier

from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score, make_scorer
from sklearn.metrics import classification_report

%matplotlib inline
```

Обзорное исследование данных.

Загрузим датасет.

```
df_columns = ['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',
 'TBG_measured',
 'TBG',
 'referral_source',
 'Class']
```

```
df = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/thyroid-disease/allhypo.data', sep = ',', na_values = '?',
                     decimal=".", header =None)
df.columns = df_columns
```

```
df.shape
```

```
(2800, 30)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2800 entries, 0 to 2799
Data columns (total 30 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   age                        2799 non-null   float64
 1   sex                        2690 non-null   object
 2   on_thyroxine               2800 non-null   object
 3   query_on_thyroxine         2800 non-null   object
 4   on_antithyroid_medication  2800 non-null   object
 5   sick                       2800 non-null   object
 6   pregnant                   2800 non-null   object
 7   thyroid_surgery            2800 non-null   object
 8   I131_treatment             2800 non-null   object
 9   query_hypothyroid          2800 non-null   object
 10  query_hyperthyroid         2800 non-null   object
 11  lithium                    2800 non-null   object
 12  goitre                     2800 non-null   object
 13  tumor                      2800 non-null   object
 14  hypopituitary              2800 non-null   object
 15  psych                      2800 non-null   object
 16  TSH_measured               2800 non-null   object
 17  TSH                        2516 non-null   float64
 18  T3_measured                2800 non-null   object
 19  T3                         2215 non-null   float64
 20  TT4_measured               2800 non-null   object
 21  TT4                        2616 non-null   float64
 22  T4U_measured               2800 non-null   object
 23  T4U                        2503 non-null   float64
 24  FTI_measured               2800 non-null   object
 25  FTI                        2505 non-null   float64
 26  TBG_measured               2800 non-null   object
 27  TBG                        0 non-null      float64
 28  referral_source            2800 non-null   object
 29  Class                      2800 non-null   object
```

Преобразуем метки классов и выделим из них диагнозы.

```
df.Class = df.Class.str.split('.').str[0]
df.Class = df.Class.str.replace(' ', '_')
```

```
df.head()
```

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	pregnant	thyroid_surgery	I131_treatment	query_h
0	41.0	F	f	f	f	f	f	f	f	
1	23.0	F	f	f	f	f	f	f	f	
2	46.0	M	f	f	f	f	f	f	f	
3	70.0	F	t	f	f	f	f	f	f	
4	70.0	F	f	f	f	f	f	f	f	

```
df.tail()
```

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	pregnant	thyroid_surgery	I131_treatment	quer
2795	70.0	M	f	f	f	f	f	f	f	
2796	73.0	M	f	t	f	f	f	f	f	
2797	75.0	M	f	f	f	f	f	f	f	
2798	60.0	F	f	f	f	f	f	f	f	
2799	81.0	F	f	f	f	f	f	f	f	

Также посмотрим какие значения содержит целевая переменная:

```
df.Class.unique()
```

```
array(['negative', 'compensated_hypothyroid', 'primary_hypothyroid',
       'secondary_hypothyroid'], dtype=object)
```

```
df.Class.value_counts()
```

```
negative                   2580
compensated_hypothyroid     154
```

Итого: есть набор данных из 30 колонок и 2800 строк, различных типов.

Часть данных - это бинарные признаки (true/false, male/female), другая часть - это численные значения, а также еще есть referral_source (WEST, STMW, SVHC, SVI, SVHD, other).

Целевая переменная Class состоит из следующих значений: ('negative', 'compensated_hypothyroid', 'primary_hypothyroid', 'secondary_hypothyroid')

Перед тем, как приступать к каким-либо преобразованиям требуется убедиться, в каких объемах есть пропуски.

Проверим, какая часть данных отсутствует, для этого выведем долю и количество пропущенных значений:

```
1 print (pd.concat([1- df.count() / df.shape[0], df.isna().sum()],axis=1))
```

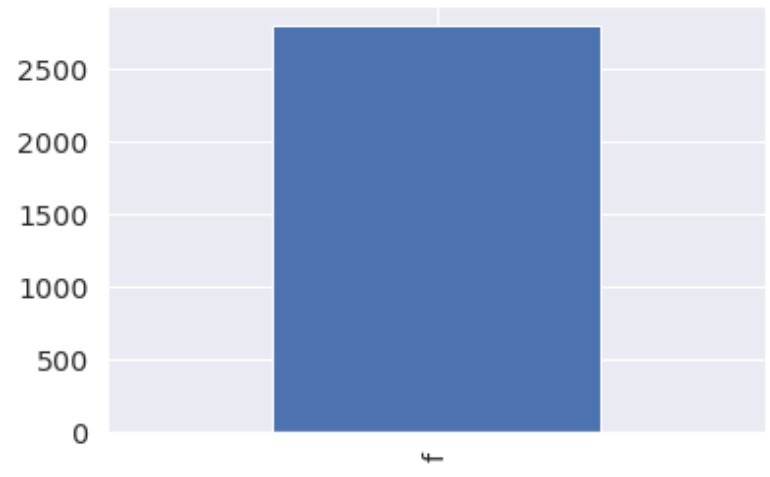
```
                                 0     1
age                       0.000357     1
sex                       0.039286   110
on_thyroxine              0.000000     0
query_on_thyroxine        0.000000     0
on_antithyroid_medication 0.000000     0
sick                      0.000000     0
pregnant                  0.000000     0
thyroid_surgery           0.000000     0
I131_treatment            0.000000     0
query_hypothyroid         0.000000     0
query_hyperthyroid        0.000000     0
lithium                   0.000000     0
goitre                    0.000000     0
tumor                     0.000000     0
hypopituitary             0.000000     0
psych                     0.000000     0
TSH_measured              0.000000     0
TSH                       0.101429   284
T3_measured               0.000000     0
T3                        0.208929   585
TT4_measured              0.000000     0
TT4                       0.065714   184
T4U_measured              0.000000     0
T4U                       0.106071   297
FTI_measured              0.000000     0
FTI                       0.105357   295
TBG_measured              0.000000     0
TBG                       1.000000  2800
referral_source           0.000000     0
Class                     0.000000     0
```

Столбец TBG состоит из пропусков полностью, поэтому его придется дропнуть.

В остальных столбцах процент пропусков не превышает 20%, что достаточно неплохо и с этим нужно работать.

```
1 df['TBG_measured'].value_counts().plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa8b06a3518>
```



Столбец TBG_measured состоит только из False, поэтому его тоже удалим.

```
1 df.drop(['TBG', 'TBG_measured'],axis=1,inplace=True)
```

Исследуем величины, в которых содержаться пропуски.

```
col_with_na = df.columns[df.isna().sum() > 0 ]
col_with_na
```

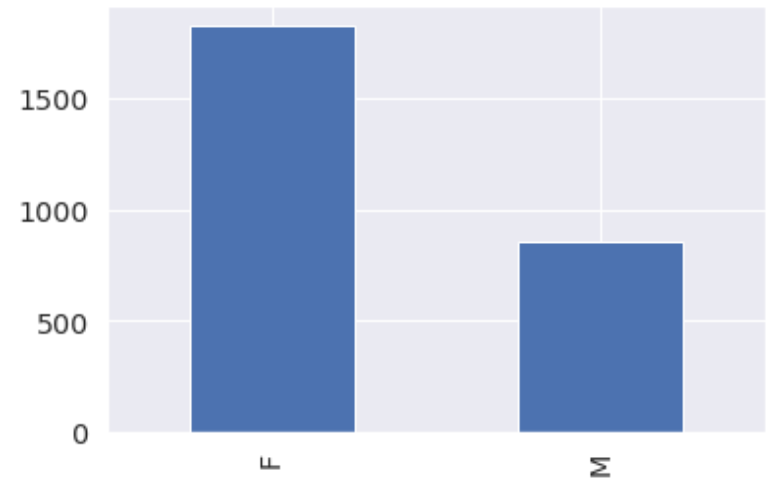
```
Index(['age', 'sex', 'TSH', 'T3', 'TT4', 'T4U', 'FTI'], dtype='object')
```

Так как 6 из 7 признаков - численные, то графики в данном случае строить смысла нет, чтобы не дублировать вывод одной и той же информации. Численные признаки будут рассматриваться далее.

Для полов построим отдельно гистограмму. В целом, для анализа пропусков этот график не имеет смысла, но обзорно посмотреть на это соотношение стоит.

```
df['sex'].value_counts().plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa8d89fe390>
```



Как оказалось, женщин в выборке в два раза больше, что говорит о дисбалансе в этом признаке.

Зафиксируем целевую переменную а также, выделим числовые и категориальные признаки в отдельные наборы данных.

```
seed = 10

target = 'Class'
numeric_columns = df.select_dtypes(include=np.number).columns.tolist()
cat_columns = df.select_dtypes(include=['object']).columns.tolist()
```

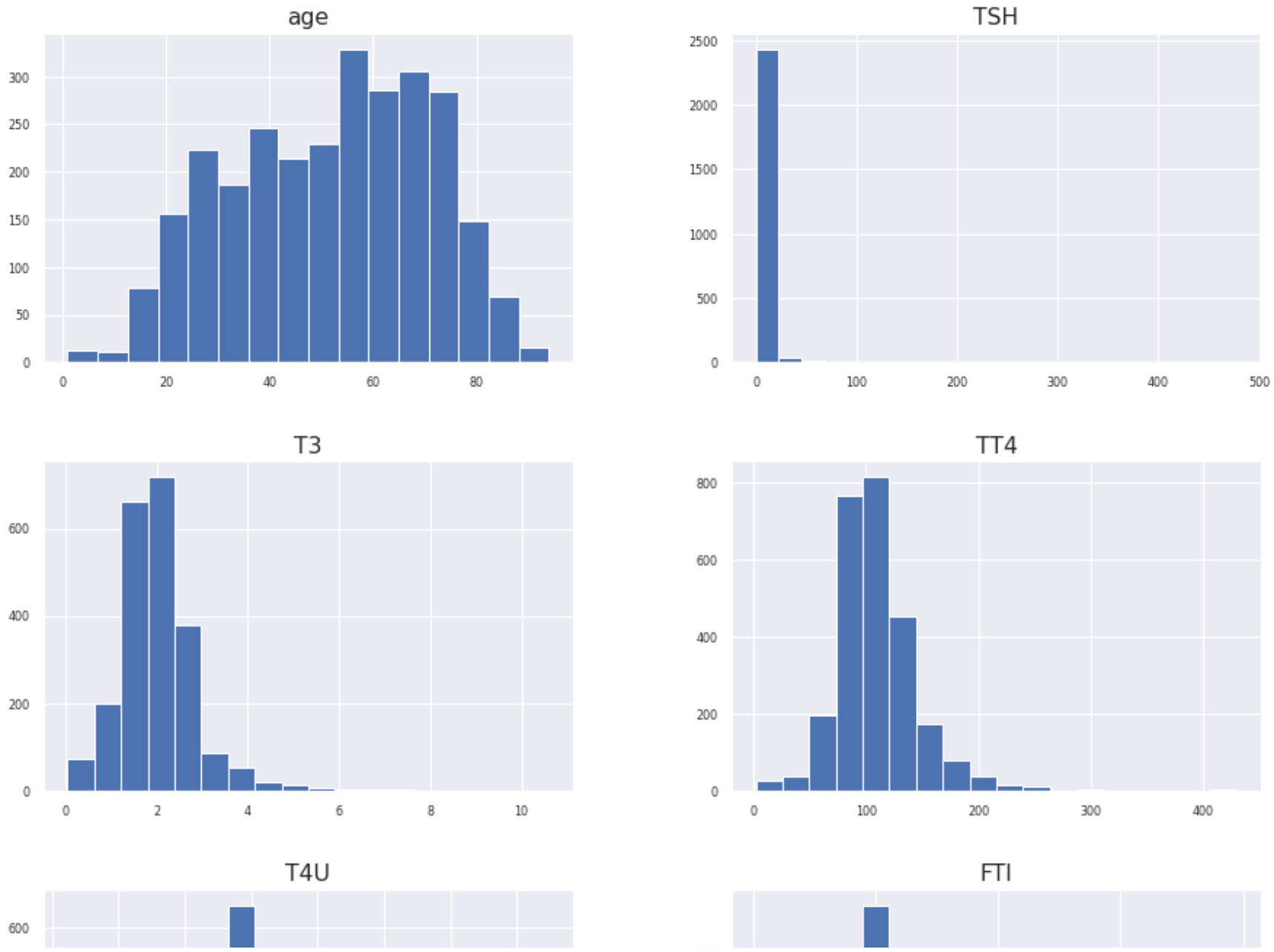
Приступим к анализу численных признаков.

Выведем численные статистики.

```
df[numeric_columns].describe()
```

	age	TSH	T3	TT4	T4U	FTI
count	2799.00000	2516.000000	2215.000000	2616.000000	2503.000000	2505.000000
mean	51.84423	4.672150	2.024966	109.072401	0.997912	110.787984
std	20.46116	21.449453	0.824600	35.392443	0.194390	32.883986
min	1.00000	0.005000	0.050000	2.000000	0.310000	2.000000
25%	36.00000	0.440000	1.600000	88.000000	0.880000	93.000000
50%	54.00000	1.400000	2.000000	104.000000	0.980000	107.000000
75%	67.00000	2.600000	2.400000	125.000000	1.080000	124.000000
max	455.00000	478.000000	10.600000	430.000000	2.120000	395.000000

```
df[numeric_columns].hist(figsize=(15, 15), bins='doane', xlabelsize=8, ylabelsize=8);
```



Вывод:

Скорее всего выбросы есть в:

- Age , т.к. не может быть возраст 455 (возможно такое и бывает, но из моего опыта, такое невозможно)
- TSH, TT4, FTI т.к. максимальные значения этих величин выше 75 процентов данных. - эти величины обозначают количество гормонов в крови.

Удалим выброс в Age, т.к. скорее всего это ошибка ввода.

Можно конечно предположить, что человек хотел ввести, 45.5 или 55 и заменить на выбранное число, но так делать некорректно, по отношению к данным.

Проверим значение данного выброса:

```
1 df[df.age > 100]
```

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	pregnant	thyroid_surgery	I131_treatment	que
1364	455.0	F	f	f	f	f	f	f	f	

```
1 df.drop(index=1364, inplace=True)
2 #сразу обновим индексы
3 df.reset_index(drop=True, inplace=True)
```

Рассмотрим величину TSH.

```
1 ind_tsh = df[df.TSH > 300].index
2 df.iloc[ind_tsh]
```

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	pregnant	thyroid_surgery	I131_treatment	que
1165	18.0	F	t	f	f	f	f	f	f	
2506	2.0	NaN	f	f	f	f	f	f	f	
2771	25.0	F	f	f	f	f	f	f	f	

Это не является выбросом, такой вывод сделан из анализа предметной области, высокий уровень TSH говорит об ослабленной щитовидной железе, что подтверждается значением переменной класс.

FTI и TT4 будем рассматривать совместно, т.к. они взаимосвязаны.

```
ind_TT4_FTI = df[(df.TT4 > 250) & (df.FTI > 250)].index
df.iloc[ind_TT4_FTI,:]
```

	age	sex	on_thyroxine	query_on_thyroxine	on_antithyroid_medication	sick	pregnant	thyroid_surgery	I131_treatment	quer
604	27.0	F	f	f	f	f	f	f	f	
743	41.0	F	f	f	f	f	f	f	f	
1414	41.0	F	f	f	f	f	t	f	f	

Исходя из предметной области, эти величины TT4 и FTI связаны между собой, the Free Thyroxine Index (FTI or T7) is obtained by multiplying the total T4 with T3 uptake.

В нашем случае T3 не uptake, но взаимосвязь прослеживается.

Посмотрим на категориальные признаки.

Перед тем, как приступать к анализу категориальных признаков столбцы 'TSH_measured', 'T3_measured', 'TT4_measured', 'T4U_measured', 'FTI_measured' дропнем, т.к. они представляют собой индикатор, было ли собрано соотвествующее численнное значение. Это не несет особой информации, т.к. в соседней колонке записано число.

```
df.drop(['TSH_measured', 'T3_measured', 'TT4_measured', 'T4U_measured', 'FTI_measured'], axis = 1, inplace=True)
```

```
# Обновим список колонок
cat_columns = df.select_dtypes(include=['object']).columns.tolist()[:-1]
```

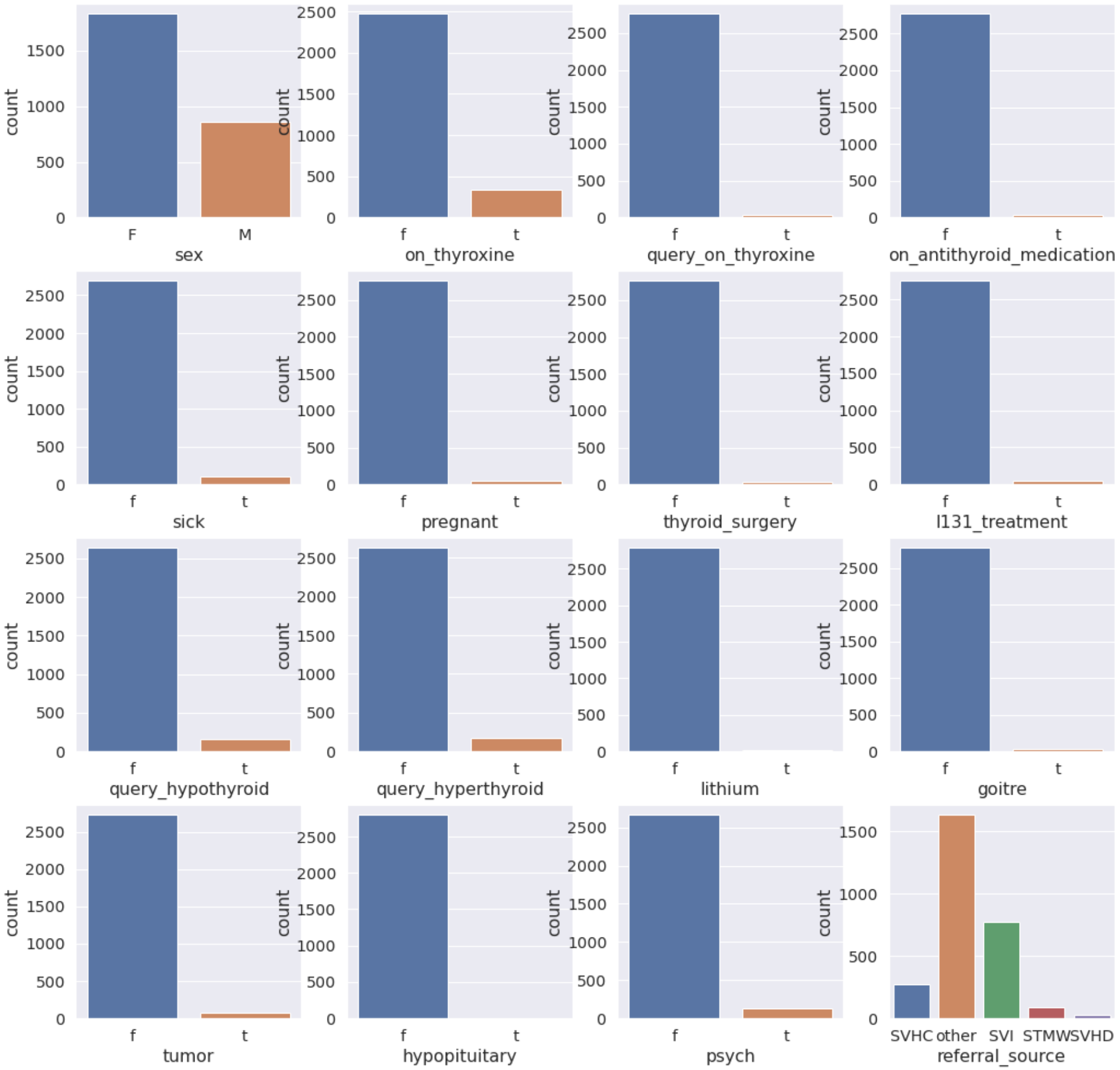
```
for i, col in enumerate(df[cat_columns].columns):
    print(df[col].value_counts())
```

```
F    1829
M     860
Name: sex, dtype: int64
f    2469
t     330
Name: on_thyroxine, dtype: int64
f    2759
t      40
Name: query_on_thyroxine, dtype: int64
f    2765
t      34
Name: on_antithyroid_medication, dtype: int64
f    2688
```

Продублируем информацию графически:

```
f    2758
```

```
plt.figure(figsize=(17, 35))
for i, col in enumerate(df[cat_columns].columns):# выводим все величины, кроме последней (целевой, ее будем рассматривать отдельно)
    plt.subplot(8,4,i+1)
    sns.countplot(x=df[col], data=df)
    plt.subplots_adjust(hspace = 0.25)
```



Из графиков и таблицы можно сделать следующий вывод:

- большинство классов несбалансированно.
- Стоит обратить внимание на признаки с thyroid в названии, возможно они могут привести к утечке данных.

Анализ целевой переменной.

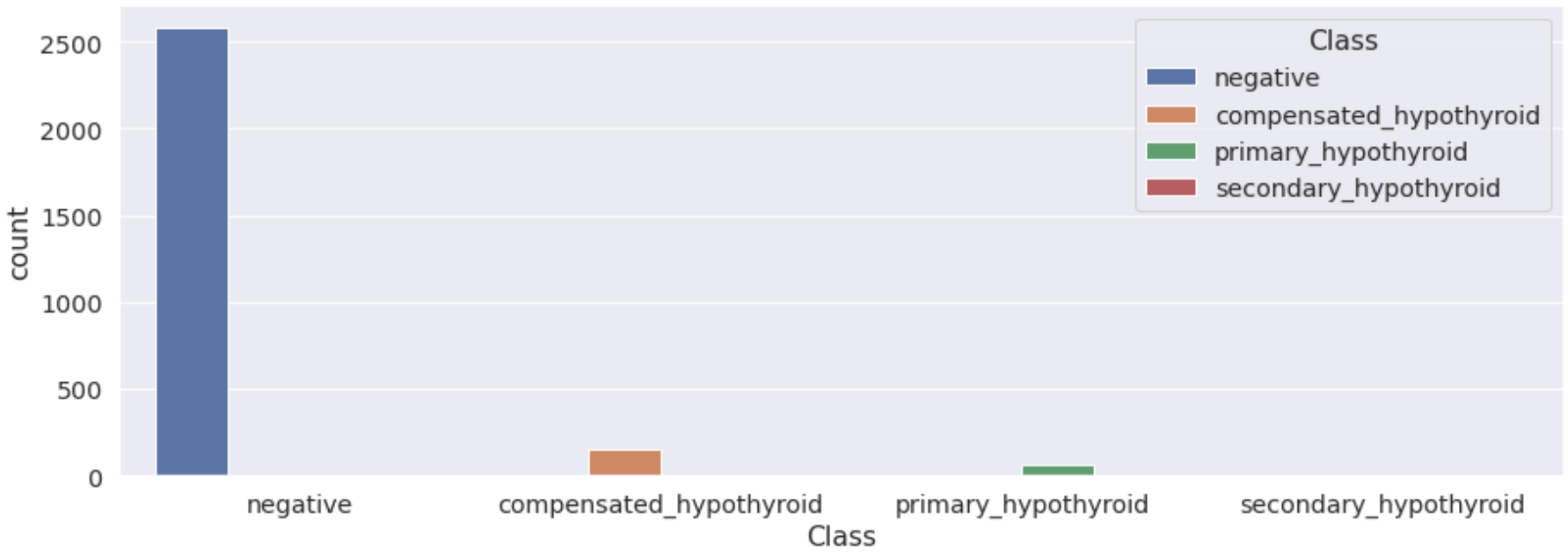
```
df[target].value_counts()
```

```
negative                   2579
compensated_hypothyroid     154
primary_hypothyroid          64
secondary_hypothyroid         2
Name: Class, dtype: int64
```

Класс secondary_hypothyroid будет несостоятельным для классификации в данной задаче, т.к. он имеет всего 2 значения.

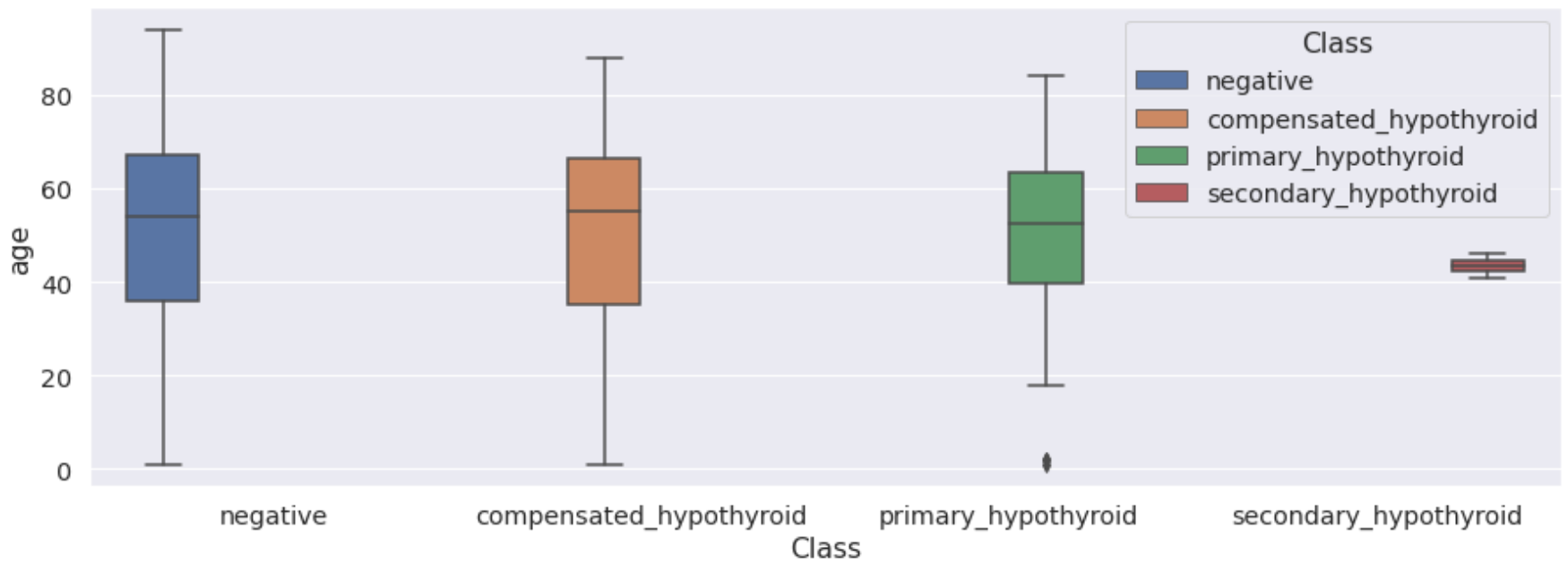
Его можно удалить, но здесь приму решение оставить, т.к. удалять целевые признаки не очень хорошо + всегда есть вероятность получить больше данных (этот комментарий не совсем корректен к этой задаче, но в плане общего подхода может позволить совершить меньше ошибок).

```
plt.figure(figsize=(15, 5))
sns.countplot(x=df[target], hue='Class', data=df)
plt.show()
```



Между классам достаточно большой разброс, на качество классификации это скорее всего повлияет отрицательно.

```
plt.figure(figsize=(15, 5))
sns.boxplot(x=df[target],y=df.age, hue='Class', data=df)
plt.show()
```



Также дополнитель посмотрим график зависимости от возраста, как видно negative/compensated/primary на одном уровне, как видно из графика, в основном пациенты были примерно одного возраста. Скорее всего age не принесет улучшения модели.

Метрики.

В работе будем использовать следующие метрики:

Confusion Matrix - для понимания, того как алгоритм справляется с классификацией.

Precision - здесь нужен, как оценка выявления правильно поставленного диагноза (т.е. считаем случаи, когда болен к отношению болен+болен ошибочно).

Recall - для выявления, что мы вычислили все случаи болезни и ничего не пропустили.

F1-score - комбинированная оценка, учитывающая предыдущие факторы.

Здесь все перечисленные оценки будут полезны.

Подготовка данных.

Разделим данные, включим стратификацию, потому, что классы несбалансированы.

```
X = df.drop(target, axis = 1)
y = df[target]
```

Производить enumerate над переменной класса не будем, т.к. это не имеет смысла в данной задаче.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
                                                    random_state=seed, shuffle=True, stratify = y)
```

```
X_train.shape
```

```
(1959, 22)
```

```
X_test.shape
```

```
(840, 22)
```

```
y_train.shape
```

```
(1959,)
```

```
y_test.shape
```

```
(840,)
```

Настроим Pipeline для обработки данных и построения моделей.

Для численных данных пропуски будем заполнять mean, скейлим как мин-макс и попробуем их приведем к нормальному распределению.

```
num_features_pipeline = Pipeline([
    ('impute', SimpleImputer(missing_values=np.nan, strategy='mean',copy = False)),
    ('scale', MinMaxScaler()),
    ('transform', QuantileTransformer(output_distribution='normal'))
])
```

```
# для категориальных колонок действием аналогично, только используем энкодер
cat_features_pipeline = Pipeline([
    ('impute', SimpleImputer(missing_values=np.nan, strategy='most_frequent', fill_value='missing', )),
    ('onehot', OneHotEncoder(handle_unknown="ignore"))
])
```

```
# создадим трансформер, который будет применять преобразования к выбранным колонкам
preprocessor = ColumnTransformer(
    transformers=[
        ('num', num_features_pipeline, numeric_columns),
        ('cat_bin', cat_features_pipeline, cat_columns)
    ]
)
```

Выбор алгоритмов.

В качестве алгоритмов были выбраны:

- логистическая регрессия, неплохой и быстрый алгоритм, который хорошо подходит для быстрого получения результатов.
- алгоритм k-ближайших соседей, выбран потому, что в основе его лежит достаточно простая гипотеза о том, что можно судить о классе объекта (исходя из выбранной метрики) по его соседям. Т.к. речь идет про диагностировании заболеваний, то, возможно это окажется эффективным способом верно предсказывать заболевание по текущему набору данных.

- алгоритм градиентного бустинга, этот алгоритм, был выбран в силу того, что является эффективным решением. Он сочетает в себе скорость работы и содержит простую, но эффективную идею того, что ансамбль слабых моделей в совокупности дает хороший результат.

Логистическая регрессия

```
# т.к. классов много, используем подход 1 против всех
clf_log = OneVsRestClassifier(LogisticRegression(multi_class = 'multinomial',
                                                 solver='newton-cg',
                                                 random_state = seed,
                                                 n_jobs = -1,
                                                 max_iter = 10e2,
                                                 class_weight='balanced'))
```

```
baseline_pipeline_logit = Pipeline(
    steps=[
        ('preprocessing', preprocessor),
        ('classify', clf_log)
    ]
)
```

```
logit_regr = baseline_pipeline_logit.fit(X_train, y_train)
```

Посчитаем метрики для этого baseline'а.

```
y_pred_train = logit_regr.predict(X_train)
y_pred_test = logit_regr.predict(X_test)
print ("on train")
print(classification_report(y_train, y_pred_train))
print("confusion matrix on train")
print(confusion_matrix(y_train, y_pred_train))
```

```
on train
                         precision    recall  f1-score   support

compensated_hypothyroid       0.66      0.96      0.78       108
               negative       1.00      0.97      0.98      1805
    primary_hypothyroid       0.70      0.78      0.74        45
  secondary_hypothyroid       0.00      0.00      0.00         1

               accuracy                           0.96      1959
              macro avg       0.59      0.68      0.63      1959
           weighted avg       0.97      0.96      0.97      1959

confusion matrix on train
[[ 104    0    4    0]
 [  43 1751   11    0]
 [  10    0   35    0]
 [   0    1    0    0]]
/usr/local/lib/python3.6/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetricWarning: Precision and F-score are
  _warn_prf(average, modifier, msg_start, len(result))
```

```
print ("on test")
print(classification_report(y_test, y_pred_test))

print("confusion matrix on test")
print(confusion_matrix(y_test, y_pred_test))
```

```
on test
                         precision    recall  f1-score   support

compensated_hypothyroid       0.66      0.96      0.78        46
               negative       1.00      0.96      0.98       774
    primary_hypothyroid       0.68      0.89      0.77        19
```

KNN

```
                macro avg       0.98      0.70      0.69       840
```

```
clf_knn = KNeighborsClassifier(n_neighbors=3, algorithm='kd_tree',weights = 'uniform')
baseline_pipeline_knn = Pipeline(
    steps=[
        ('preprocessing', preprocessor),
        ('classify', clf_knn)
    ]
)
```

```
knn = baseline_pipeline_knn.fit(X_train, y_train)
```

```
y_pred_train = knn.predict(X_train)
y_pred_test = knn.predict(X_test)
print ("on train")
print(classification_report(y_train, y_pred_train))
print("confusion matrix on train")
print(confusion_matrix(y_train, y_pred_train))
```

```
on train
                         precision    recall  f1-score   support

compensated_hypothyroid       0.91      0.69      0.79       108
               negative       0.98      1.00      0.99      1805
    primary_hypothyroid       0.93      0.89      0.91        45
  secondary_hypothyroid       0.00      0.00      0.00         1

               accuracy                           0.98      1959
              macro avg       0.71      0.64      0.67      1959
           weighted avg       0.98      0.98      0.97      1959

confusion matrix on train
[[  75   32    1    0]
 [   5 1798    2    0]
 [   2    3   40    0]
 [   0    1    0    0]]
/usr/local/lib/python3.6/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetricWarning: Precision and F-score are
  _warn_prf(average, modifier, msg_start, len(result))
```

```
print ("on test")
print(classification_report(y_test, y_pred_test))

print("confusion matrix on test")
print(confusion_matrix(y_test, y_pred_test))
```

```
on test
                         precision    recall  f1-score   support

compensated_hypothyroid       0.66      0.41      0.51        46
               negative       0.96      0.99      0.98       774
    primary_hypothyroid       0.93      0.74      0.82        19
  secondary_hypothyroid       0.00      0.00      0.00         1

               accuracy                           0.95       840
              macro avg       0.64      0.53      0.58       840
           weighted avg       0.94      0.95      0.95       840

confusion matrix on test
[[ 19  27   0   0]
 [  7 766   1   0]
 [  3   2  14   0]
 [  0   1   0   0]]
/usr/local/lib/python3.6/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetricWarning: Precision and F-score are
  _warn_prf(average, modifier, msg_start, len(result))
```

Ситуация кардинально не изменилась. compensated_hypothyroid стал распознаваться хуже, primary_hypothyroid стал распознаваться лучше.

Gradient Boosting

```
clf_gb = GradientBoostingClassifier(n_estimators=20, learning_rate=0.5, max_depth=5, random_state=0)
baseline_pipeline_gb = Pipeline(
    steps=[
        ('preprocessing', preprocessor),
        ('classify', clf_gb)
    ]
)
```

```
gb = baseline_pipeline_gb.fit(X_train, y_train)
```

```
y_pred_train = gb.predict(X_train)
y_pred_test = gb.predict(X_test)
print ("on train")
print(classification_report(y_train, y_pred_train))
print("confusion matrix on train")
print(confusion_matrix(y_train, y_pred_train))
```

```
on train
                          precision    recall  f1-score   support

compensated_hypothyroid       1.00      1.00      1.00       108
               negative       1.00      1.00      1.00      1805
    primary_hypothyroid       1.00      1.00      1.00        45
  secondary_hypothyroid       1.00      1.00      1.00         1

               accuracy                           1.00      1959
              macro avg       1.00      1.00      1.00      1959
           weighted avg       1.00      1.00      1.00      1959

confusion matrix on train
[[ 108    0    0    0]
 [   0 1805    0    0]
 [   0    0   45    0]
 [   0    0    0    1]]
```

```
print ("on test")
print(classification_report(y_test, y_pred_test))

print("confusion matrix on test")
print(confusion_matrix(y_test, y_pred_test))
```

```
on test
                          precision    recall  f1-score   support

compensated_hypothyroid       0.94      0.96      0.95        46
               negative       1.00      1.00      1.00       774
    primary_hypothyroid       0.85      0.89      0.87        19
  secondary_hypothyroid       0.00      0.00      0.00         1

               accuracy                           0.99       840
              macro avg       0.70      0.71      0.70       840
           weighted avg       0.99      0.99      0.99       840

confusion matrix on test
[[ 44   1   1   0]
 [  1 771   2   0]
 [  2   0  17   0]
 [  0   1   0   0]]
/usr/local/lib/python3.6/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetricWarning: Precision and F-score are
  _warn_prf(average, modifier, msg_start, len(result))
```

А вот алгоритм на основе ансамбля решающих деревьев показывает намного лучшие результаты.

Заключение

Среди сравненных классфикаторов и параметров наилучшими оценками обладают (в порядке убывания):

- градиентный бустинг на деревьях принятия решений
- KNN
- логистическая регрессия

В данной работы мы провели:

- работу с датасетом, а именно - отбор и фильтрацию признаков
- исследование данных при помощи статистик и графиков
- подготовку данных перед подачей их в алгоритмы
- обучение нескольких алгоритмов машинонного обучения
- сравнение их результатов