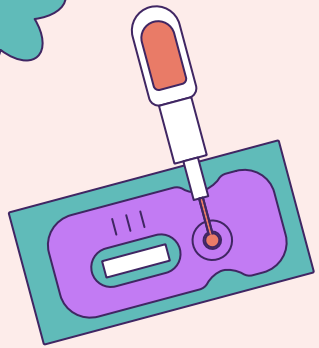
ST1505 DevOps & AI Automations : Covid, Flu, Cold & Allergy Prediction

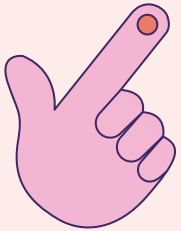
Soh Hong Yu | P2100775 | DAAA/FT/2B/01

TABLE OF CONTENTS



01

BACKGROUND INFORMATION



COLD, FLU & ALLERGIES

In Singapore, due to our humid environment, many people get cold and flu. In 2021, 588 deaths caused by flu and the cold in Singapore alone. Furthermore, 24% of Singaporeans are allergic to pollen. As the symptoms of all 3 illness are very similar, knowing which one you have is very crucial especially if it is very serious.

COVID-19

During Covid-19, Many Singaporeans despite having a small symptoms like cough and sneezing, they will go to their nearby hospitals and general clinics to check if they have been infected by Covid-19. This increase the workload of the hospitals and general clinics. This prompted the Singaporean Government to launch campaigns to promote and educate citizens on what they can do when they experience symptoms similar to Covid-19.

PROJECT OBJECTIVE

REDUCE WORK

Doctors and Nurse will only need to provide the necessary care to the other patients that is suffering a more severe illness

SELF TESTING

Patients will have a piece of mind and would be educated based. This will also allow them to monitor themselves.

Presenting To You...

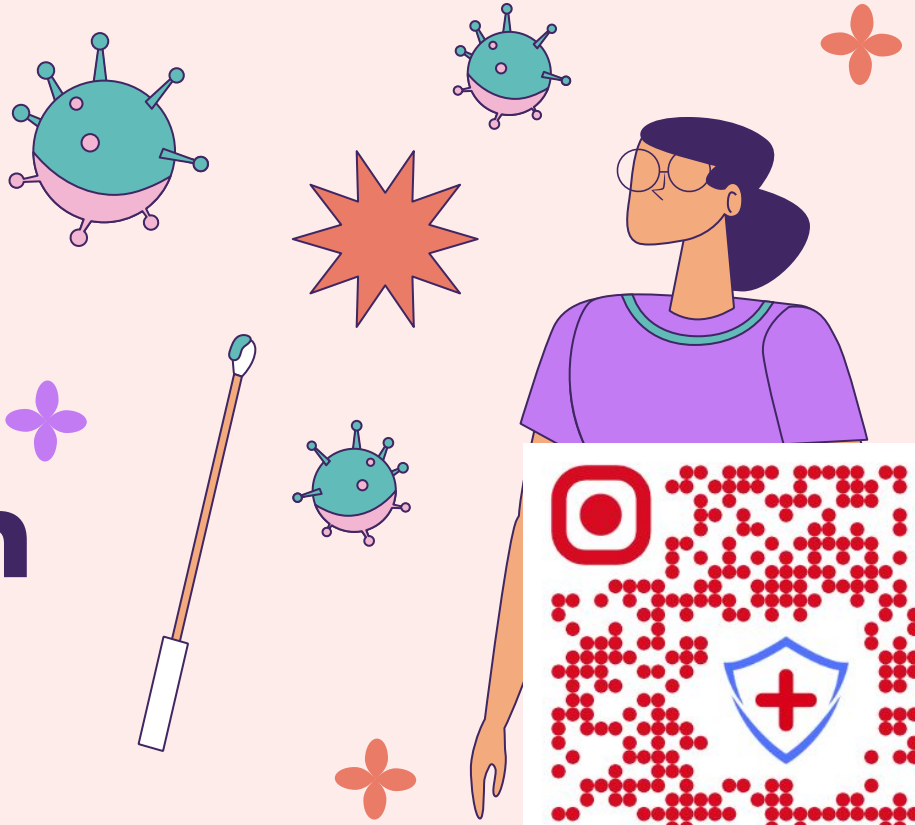
Immune is a simple to use application that helps users diagnose what illness does the user have through the present of symptoms. And AI model will be used to predict the illness and from there, the application will advise users on what to do next!

IMMUNE

Goodbye Illness, Hello Health

02

Application Demo



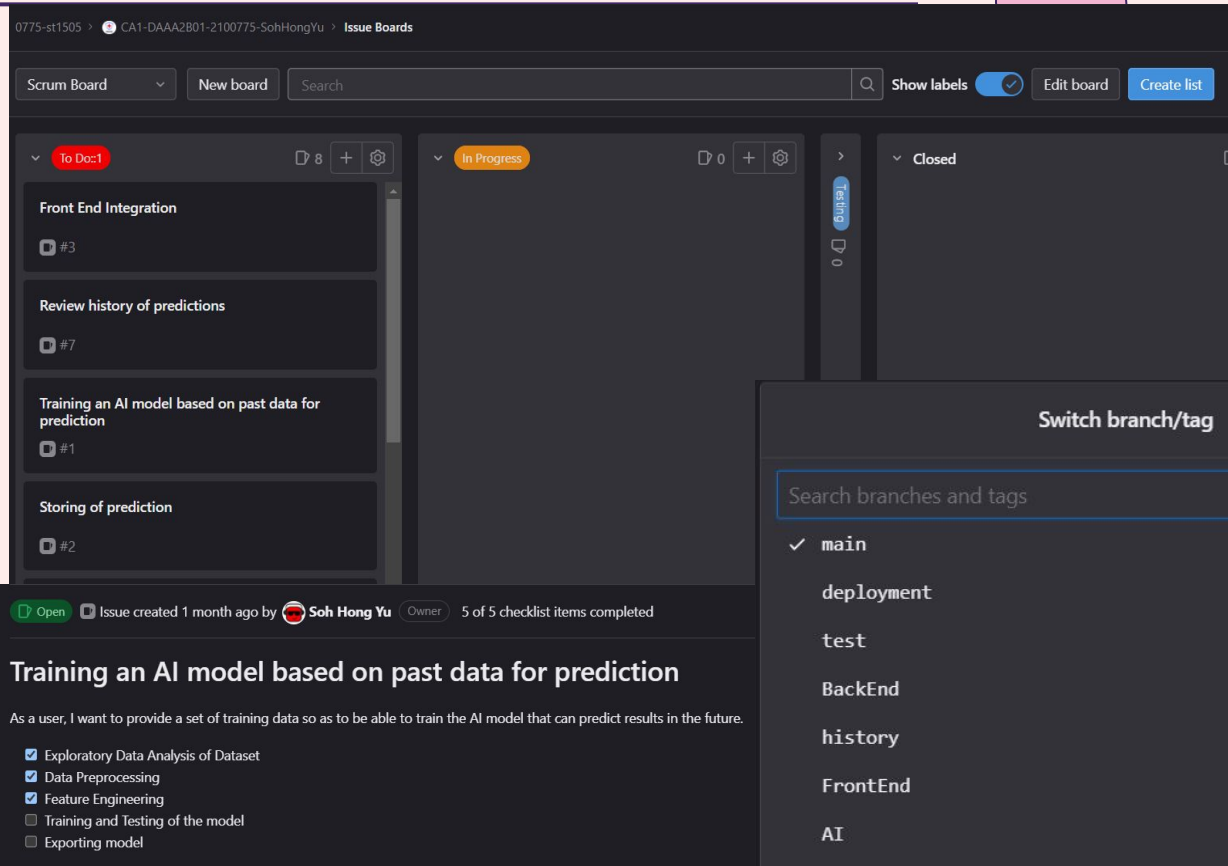
03

DevOps Process

SETTING UP

Before developing the application, we need to do some scrum boarding. This allows us to keep track of our application development. [To Do, In Progress and Testing Labels]

After doing the scrum board, it is time to make branches. This allow us to make changes to the folder in a controlled environment. [No of Branches: 6]

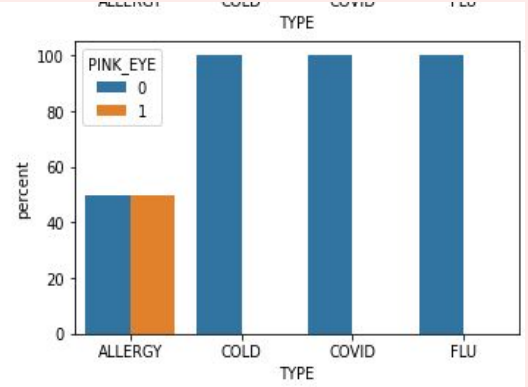


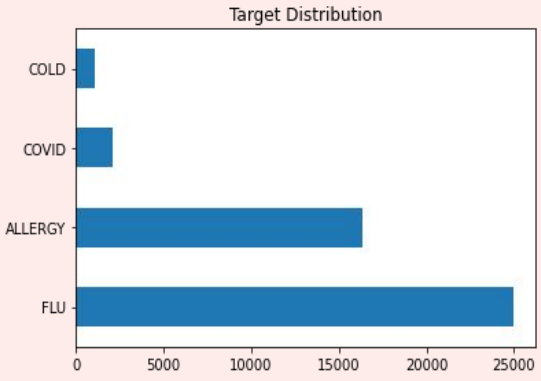
DATASET

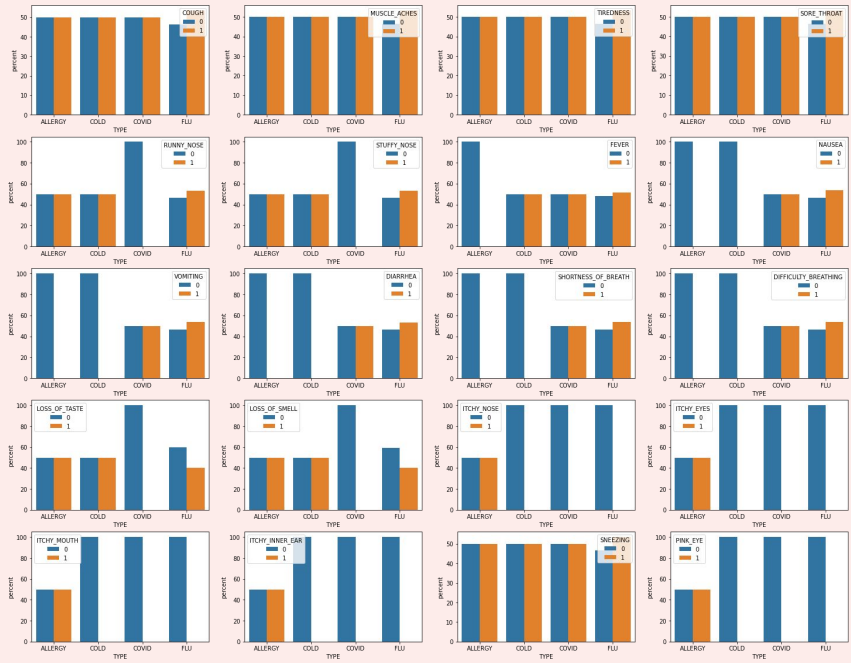
COUGH : Boolean value that indicates if coughing is a symptom.
MUSCLE_ACHES : Boolean value that indicates if muscle aches is a symptom.
TIREDNESS : Boolean value that indicates if tiring is a symptom.
SORE_THROAT : Boolean value that indicates if sore throat is a symptom.
RUNNY_NOSE : Boolean value that indicates if running nose is a symptom.
STUFFY_NOSE : Boolean value that indicates if stuffy nose is a symptom.
FEVER : Boolean value that indicates if fever is a symptom.
NAUSEA : Boolean value that indicates if nausea is a symptom.
VOMITING : Boolean value that indicates if vomiting is a symptom.
DIARRHEA : Boolean value that indicates if diarrhea is a symptom.
SHORTNESS_OF_BREATH : Boolean value that indicates if shortness of breath is a symptom.
DIFFICULTY_BREATHING : Boolean value that indicates if difficulty breathing is a symptom.
LOSS_OF_TASTE : Boolean value that indicates if loss of taste is a symptom.
LOSS_OF_SMELL : Boolean value that indicates if loss of smell is a symptom.
ITCHY_NOSE : Boolean value that indicates if itchy nose is a symptom.
ITCHY_EYES : Boolean value that indicates if itchy eyes is a symptom.
ITCHY_MOUTH : Boolean value that indicates if itchy mouth is a symptom.
ITCHY_INNER_EAR : Boolean value that indicates if itchy inner ear is a symptom.
SNEEZING : Boolean value that indicates if sneezing is a symptom.
PINK_EYE : Boolean value that indicates if pink eye is a symptom.
TYPE : Type of illness [flu, cold, allergy & covid].

EXPLORATORY DATA ANALYSIS

We can see that the target distribution is not equal and therefore, we will be doing some preprocessing later. We also note that from the graphs [Percentage of symptoms based on TYPE] certain symptoms are key factors. For example, Pink Eye is a key symptom of Allergy. This suggest that a decision tree based model is strong as only certain factors will lead to specific illness







DATA PREPROCESSING

1. Train - Test Split

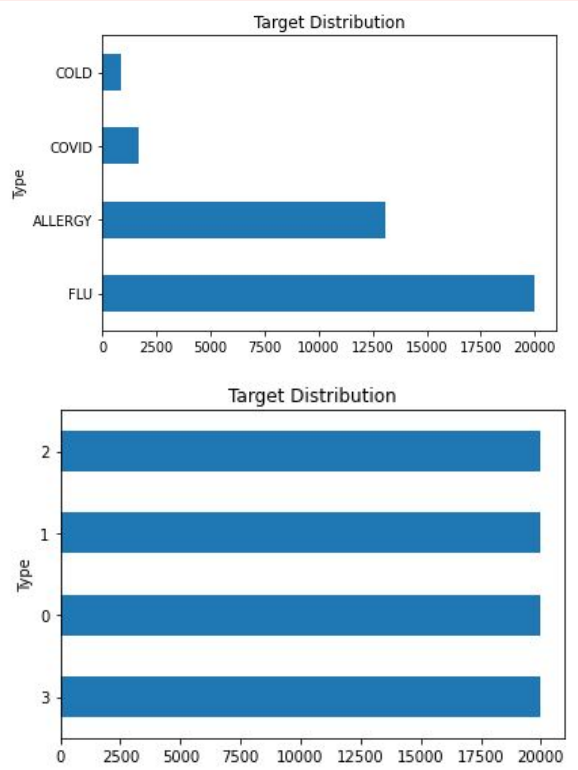
```
x_train, x_test, y_train, y_test = train_test_split(
    x, y, test_size=0.2, stratify=y, shuffle=True, random_state=42
)
x_training = x_train.copy()
```

2. Categorical Encoding [Encoding Labels]

```
encoder = LabelEncoder()
y_train_enc = encoder.fit_transform(y_train)
y_train_enc
```

3. Rebalancing of data

```
resample = RandomOverSampler(random_state=11)

X_resampled_train, y_resampled_train = resample.fit_resample(
    x_training, y_train_enc)
X_resampled_train
```



MODELLING

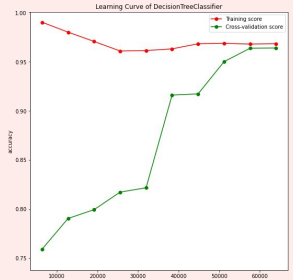
Scoring - Accuracy, Recall, F1 score & ROC_AUC

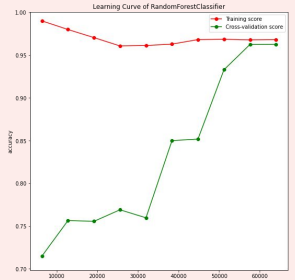
Models Used:

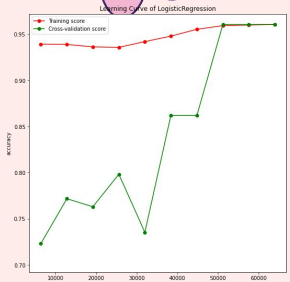
1. DummyClassifer [Baseline]
2. DecisionTreeClassifier
3. RandomForestClassifier
4. LogisticRegression
5. KNeighborsClassifier
6. ExtraTreesClassifier
7. GaussianNB

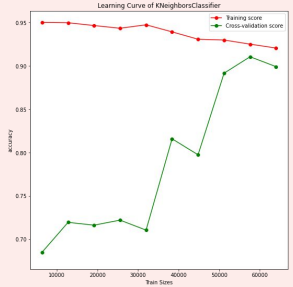
Model using cross_validation performs well at 96.3%.

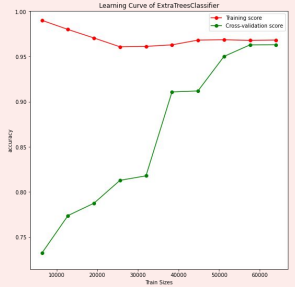
KNN model is the worse as the data is mostly booleans and it is hard to form clusters.

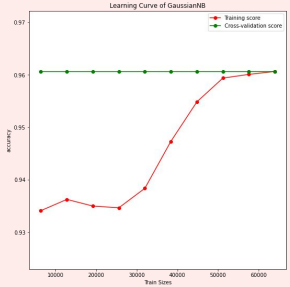










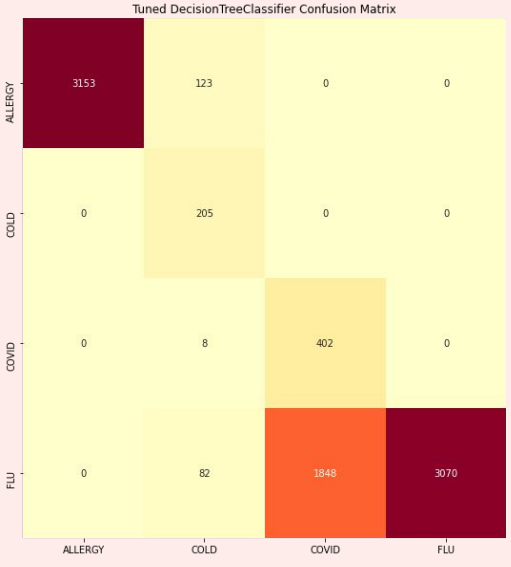


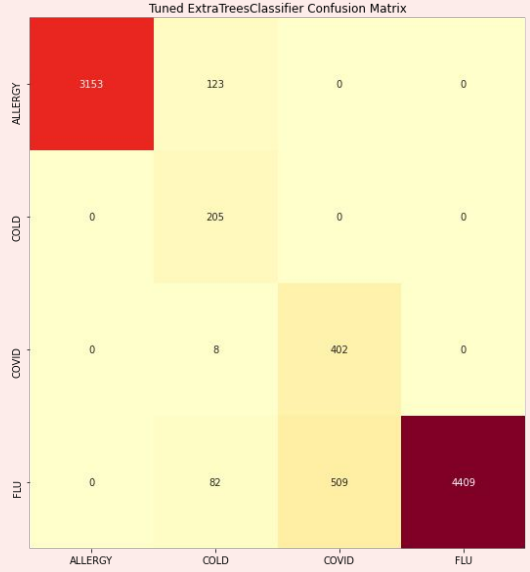
	fit_time	score_time	test_accuracy	train_accuracy	test_recall_weighted	train_recall_weighted	test_f1_weighted	train_f1_weighted	test_roc_auc_ovr_weighted	train_roc_auc_ovr_weighted
DecisionTreeClassifier	0.420508	0.042110	0.963887	0.968292	0.963887	0.968292	0.963878	0.968317	0.977634	0.994982
ExtraTreesClassifier	13.467522	0.770004	0.963125	0.968292	0.963125	0.968292	0.963108	0.968317	0.979350	0.994982
RandomForestClassifier	12.506521	0.709096	0.962662	0.968292	0.962662	0.968292	0.962643	0.968320	0.979590	0.994926
LogisticRegression	8.955542	0.047871	0.960613	0.960613	0.960613	0.960613	0.960648	0.960651	0.989725	0.989960
GaussianNB	0.160938	0.096258	0.960613	0.960613	0.960613	0.960613	0.960648	0.960651	0.989038	0.989052
KNeighborsClassifier	0.175395	110.792156	0.900350	0.921371	0.900350	0.921371	0.898684	0.920453	0.970256	0.985239

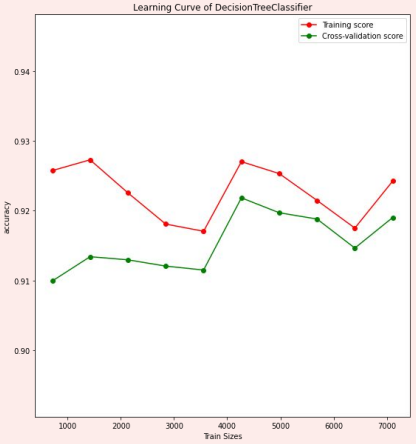
HYPERPARAMETER TUNING

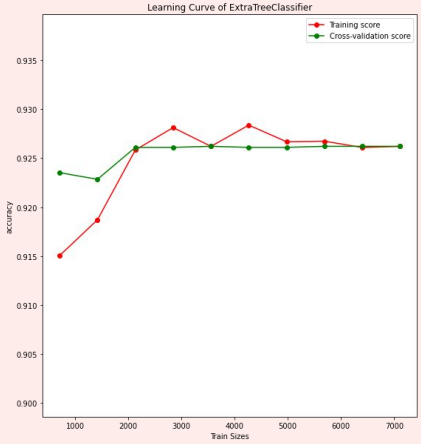
We will be hyperparameter tuning DecisionTreeClassifier and ExtraForestClassifier [Best Models from previous experiment] to see the improvement in the accuracy after tuning. We also will need to tune these models as tree based models like to over fit and therefore we will be cutting the tree and the number of branches it can form to reduce overfitting.

```
Fitting 15 folds for each of 66 candidates, totalling 990 fits
DecisionTreeClassifier(criterion='entropy', max_depth=10, max_leaf_nodes=15,
                       random_state=42)
{'max_depth': 10, 'max_leaf_nodes': 15}
0.8874499725599179

Fitting 5 folds for each of 50 candidates, totalling 250 fits
ExtraTreesClassifier(criterion='entropy', max_depth=90, max_leaf_nodes=15,
                     n_estimators=1000, random_state=42)
{'n_estimators': 1000, 'max_leaf_nodes': 15, 'max_depth': 90}
0.9571250000000001
```





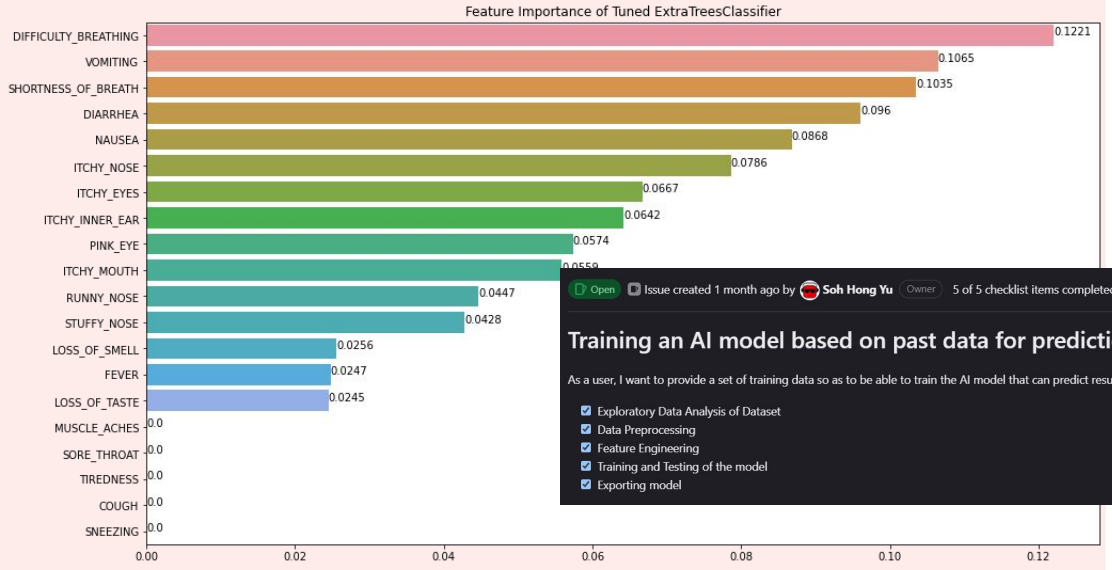




EVALUATE MODEL

From our quick classification report, we can see that ExtraTreesClassifier models are more accurate in predicting whether what type of illness the patient has.

```
pickle.dump(tuned_e_tree_clf, open("extra_trees_classifier.p", "wb"))
tuned_e_tree_clf = pickle.load(open("extra_trees_classifier.p", "rb"))
```



Open · Issue created 1 month ago by Soh Hong Yu · Owner · 5 of 5 checklist items completed

Training an AI model based on past data for prediction

As a user, I want to provide a set of training data so as to be able to train the AI model that can predict results in the future.

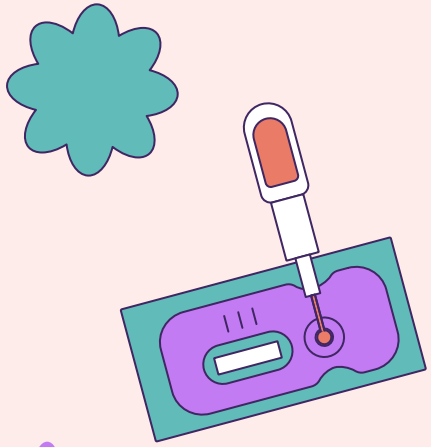
- [x] Exploratory Data Analysis of Dataset
- [x] Data Preprocessing
- [x] Feature Engineering
- [x] Training and Testing of the model
- [x] Exporting model

DecisionTreeClassifier:

	precision	recall	f1-score	support
0	1.000	0.962	0.981	3276
1	0.490	1.000	0.658	205
2	0.179	0.980	0.302	410
3	1.000	0.614	0.761	5000
accuracy			0.768	8891
macro avg	0.667	0.889	0.676	8891
weighted avg	0.950	0.768	0.818	8891

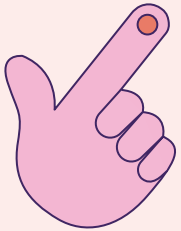
ExtraTreesClassifier:

	precision	recall	f1-score	support
0	1.000	0.962	0.981	3276
1	0.490	1.000	0.658	205
2	0.441	0.980	0.609	410
3	1.000	0.882	0.937	5000
accuracy			0.919	8891
macro avg	0.733	0.956	0.796	8891
weighted avg	0.962	0.919	0.932	8891



04

TESTING & ADVANCED



UNEXPECTED FAILURE TESTING

We made an unexpected failure function to test if there is unexpected failures. Looks like after running the FC,

```
tests/test_application.py::test_EntryClass[predictionList0] [1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1]
PASSED
tests/test_application.py::test_EntryClass[predictionList1] [1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1]
PASSED
tests/test_application.py::test_EntryClass[predictionList2] [1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2, 1]
PASSED
tests/test_application.py::test_EntryClass[predictionList3] [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1]
PASSED
```

```python
@pytest.mark.parametrize("predictionList", [
    [1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1],
    [1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1],
    [1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2, 1],
    [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1]
])
# 3: Write the test function pass in the arguments
def test_EntryClass(predictionList, capsys):
    with capsys.disabled():
        print(predictionList)
        now = datetime.datetime.utcnow()
        new_entry = Entry(…
        assert new_entry.cough == predictionList[0]
        assert new_entry.muscle_aches == predictionList[1]
        assert new_entry.tiredness == predictionList[2]
        assert new_entry.sore_throat == predictionList[3]
        assert new_entry.runny_nose == predictionList[4]
        assert new_entry.stuffy_nose == predictionList[5]
        assert new_entry.fever == predictionList[6]
        assert new_entry.nausea == predictionList[7]
        assert new_entry.vomiting == predictionList[8]
        assert new_entry.diarrhea == predictionList[9]
        assert new_entry.shortness_of_breath == predictionList[10]
        assert new_entry.difficulty_breathing == predictionList[11]
        assert new_entry.loss_of_taste == predictionList[12]
        assert new_entry.loss_of_smell == predictionList[13]
        assert new_entry.itchy_nose == predictionList[14]
        assert new_entry.itchy_eyes == predictionList[15]
        assert new_entry.itchy_mouth == predictionList[16]
        assert new_entry.itchy_inner_ear == predictionList[17]
        assert new_entry.sneezing == predictionList[18]
        assert new_entry.pink_eye == predictionList[19]
        assert new_entry.prediction == predictionList[20]
        assert new_entry.predicted_on == now
```

EXPECTED FAILURE TESTING

We will be using the same function as the unexpected failure testing. This allows us to moderate and check for any of the data has any outliers that was not validated.

```
PASSED
tests/test_application.py::test_EntryValidation[predictionList0] [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, -1]
XPASS (arguments <= 0)
tests/test_application.py::test_EntryValidation[predictionList1] [-1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2]
XPASS (arguments <= 0)
tests/test_application.py::test_EntryValidation[predictionList2] [1, -1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2]
XPASS (arguments <= 0)
tests/test_application.py::test_EntryValidation[predictionList3] [1, 1, -1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2]
XPASS (arguments <= 0)
tests/test_application.py::test_EntryValidation[predictionList4] [1, 1, 1, -1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2]
XPASS (arguments <= 0)
tests/test_application.py::test_EntryValidation[predictionList5] [1, 1, 1, 1, -1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2]
XPASS (arguments <= 0)
tests/test_application.py::test_EntryValidation[predictionList6] [1, 1, 1, 1, 1, -1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2]
XPASS (arguments <= 0)
```

```python
@pytest.mark.xfail(reason="arguments <= 0")
@pytest.mark.parametrize("predictionList", [
    [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, -1],
    [-1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, -1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, -1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, -1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, -1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, -1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, -1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, 0, -1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, 0, 0, -1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, 0, 0, 0, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, -1, 1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, -1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, -1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, -1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, -1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, -1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, -1, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, -1, 0, 2],
    [2, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 2, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 2, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 2, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 2, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 2, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, 2, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, 0, 2, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, 0, 0, 2, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, 0, 0, 0, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 0, 2],
    [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 0, 2],
    [1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2],
])
def test_EntryValidation(predictionList, capsys):
    test_EntryClass(predictionList, capsys)
```

CONSISTENCY TESTING

We group arrays into 3 array of random index. The array in an array will be looped through, then the values are feed into the AI Model to test if the output prediction is consistent.

```
tests/test_application.py::test_deleteAPI[predictionList2] PASSED
tests/test_application.py::test_predictAPI[bigPredictionList0] PASSED
tests/test_application.py::test_predictAPI[bigPredictionList1] PASSED
tests/test_application.py::test_predictAPI[bigPredictionList2] PASSED
tests/test_auth.py::test_signUpAPI[logInInfo0] XFAIL (Already exist)
```

```python
@pytest.mark.parametrize("bigPredictionList", [
    [[1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1],
     [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1],
     [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1], ],
    [[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
     [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
     [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], ],
    [[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1],
        [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1],
        [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1], ]
])
def test_predictAPI(client, bigPredictionList, capsys):
    predictOutput = []
    for predictionList in bigPredictionList:
        with capsys.disabled():
            # prepare the data into a dictionary
            predictData = {
                "cough": predictionList[0],
                "muscle_aches": predictionList[1],
                "tiredness": predictionList[2],
                "sore_throat": predictionList[3],
                "runny_nose": predictionList[4],
                "stuffy_nose": predictionList[5],
                "fever": predictionList[6],
                "nausea": predictionList[7],
                "vomiting": predictionList[8],
                "diarrhea": predictionList[9],
                "shortness_of_breath": predictionList[10],
                "difficulty_breathing": predictionList[11],
                "loss_of_taste": predictionList[12],
                "loss_of_smell": predictionList[13],
                "itchy_nose": predictionList[14],
                "itchy_eyes": predictionList[15],
                "itchy_mouth": predictionList[16],
                "itchy_inner_ear": predictionList[17],
                "sneezing": predictionList[18],
                "pink_eye": predictionList[19],
                "id": predictionList[-1]  # User ID
            }
            response = client.post('/api/predict',
```

VALIDATION FAILURE TESTING

We check if there is any issue or validation error in the parameter. We also validate and ensure the data is inside of the database beforehand.

```
/test_application.py::test_predictAPI[bigPredictionList2] PASSED
/test_auth.py::test_signUpAPI[logInInfo0] XPASS (Not Valid Username or Password)
/test_auth.py::test_signUpAPI[logInInfo1] XFAIL (Not Valid Username or Password)
/test_auth.py::test_signUpAPI[logInInfo2] XFAIL (Not Valid Username or Password)
/test_auth.py::test_signUpAPI[logInInfo3] XFAIL (Not Valid Username or Password)
```

```python
# Validation Test
@pytest.mark.xfail(reason="Not Valid Username or Password")
@pytest.mark.parametrize("logInInfo", [
  ["sohhongyu@gmail.com","123", 0],
  ["sohhongyu123@gmail.com", "123", 1],
  ["sohhongyu@gmail.com","123123", 1],
  ["devops@gmail.com", "123", 1]
]
)
def test_signUpAPI(client, logInInfo, capsys):
    with capsys.disabled():
        # prepare the data into a dictionary
        logInData = {
            "email": logInInfo[0],
            "password": logInInfo[1]
        }
    response = client.post('/api/login',
                           data=json.dumps(logInData),
                           content_type="application/json",)
    # check the outcome of the action
    assert response.status_code == 200
    assert response.headers["Content-Type"] == "application/json"
    response_body = json.loads(response.get_data(as_text=True))
    assert not response_body["isLogin"] == logInInfo[2]
```

ENDPOINT API TESTING

To add a new endpoint, each columns of the list is used to help make the data. Once the data is done, it will be added into the database.

```
tests/test_application.py::test_addAPI[predictionList0] PASSED
tests/test_application.py::test_addAPI[predictionList1] PASSED
tests/test_application.py::test_addAPI[predictionList2] PASSED
tests/test_application.py::test_addAPI[predictionList3] PASSED
```

```python
@pytest.mark.parametrize("predictionList", [
    [1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1],
    [1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1],
    [1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2, 1],
    [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1]])
def test_addAPI(client, predictionList, capsys):
    with capsys.disabled():
        # prepare the data into a dictionary
        predictData = {···
    response = client.post('/api/add',
                           data=json.dumps(predictData),
                           content_type="application/json",)
    # check the outcome of the action
    assert response.status_code == 200
    assert response.headers["Content-Type"] == "application/json"
    response_body = json.loads(response.get_data(as_text=True))
    assert response_body["id"]
```

ENDPOINT API TESTING

Created a test case to get endpoint by index. It will take in data from the other columns to ensure all are correct.

```
tests/test_application.py::test_getAPI[predictionList0] PASSED
tests/test_application.py::test_getAPI[predictionList1] PASSED
tests/test_application.py::test_getAPI[predictionList2] PASSED
tests/test_application.py::test_deleteAPI[predictionList0] PASSED
tests/test_application.py::test_deleteAPI[predictionList1] PASSED
tests/test_application.py::test_deleteAPI[predictionList2] PASSED
```

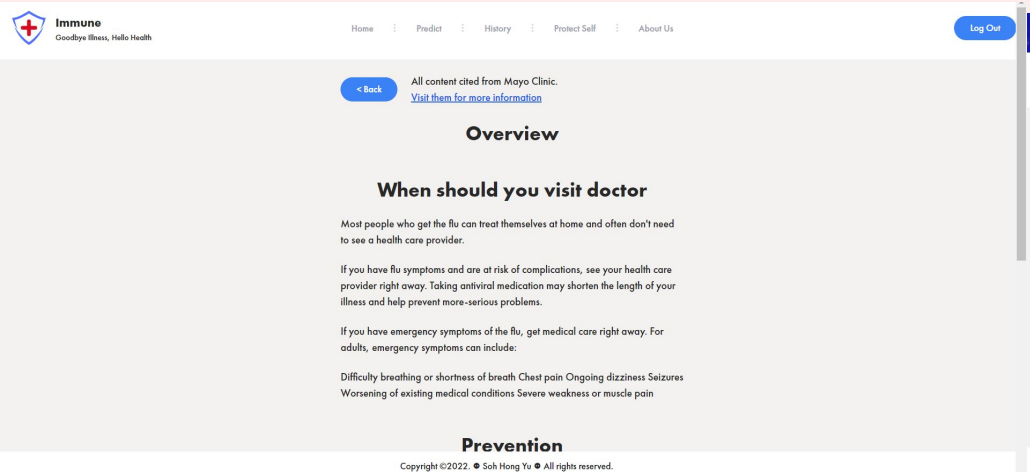
```python
@pytest.mark.parametrize("predictionList", [
    [1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1],
    [1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2, 1],
    [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1]
])
def test_deleteAPI(client, predictionList, capsys):
    with capsys.disabled():
        predictData = {···
        response = client.post(
            '/api/add', data=json.dumps(predictData), content_type="application/json",)
        response_body = json.loads(response.get_data(as_text=True))
        assert response_body["id"]
        id = response_body["id"]
        response2 = client.get(f'/api/delete/{id}')
        ret = json.loads(response2.get_data(as_text=True))

        # check the outcome of the action
        assert response2.status_code == 200
        assert response2.headers["Content-Type"] == "application/json"
        response2_body = json.loads(response2.get_data(as_text=True))
        assert response2_body["result"] == "ok"
```

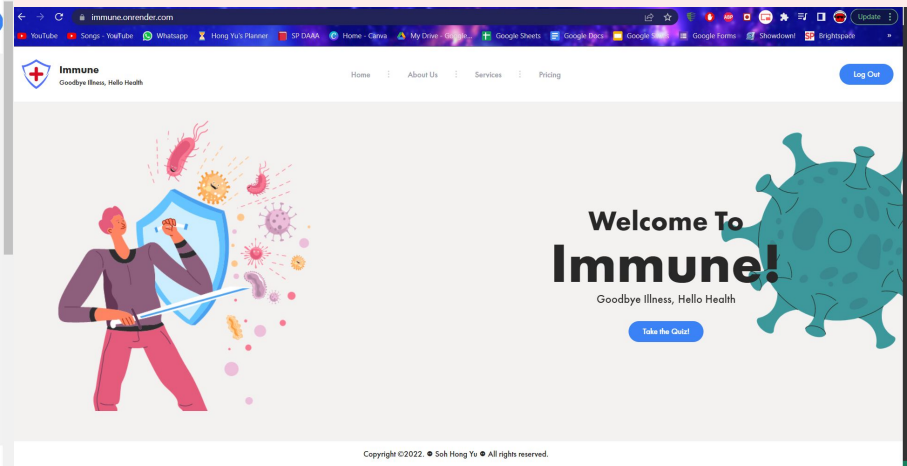
```python
@pytest.mark.parametrize("predictionList", [
    [1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1],
    [1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2],
    [1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 10],
])
def test_getAPI(client, predictionList, capsys):
    with capsys.disabled():
        response = client.get(f'/api/get/{predictionList[-1]}')
        ret = json.loads(response.get_data(as_text=True))
        # check the outcome of the action
        assert response.status_code == 200
        assert response.headers["Content-Type"] == "application/json"
        response_body = json.loads(response.get_data(as_text=True))
        assert response_body["id"] == predictionList[-1]
        assert response_body["cough"] == predictionList[0]
        assert response_body["muscle_aches"] == predictionList[1]
        assert response_body["tiredness"] == predictionList[2]
        assert response_body["sore_throat"] == predictionList[3]
        assert response_body["runny_nose"] == predictionList[4]
        assert response_body["stuffy_nose"] == predictionList[5]
        assert response_body["fever"] == predictionList[6]
        assert response_body["nausea"] == predictionList[7]
        assert response_body["vomiting"] == predictionList[8]
        assert response_body["diarrhea"] == predictionList[9]
        assert response_body["shortness_of_breath"] == predictionList[10]
        assert response_body["difficulty_breathing"] == predictionList[11]
        assert response_body["loss_of_taste"] == predictionList[12]
        assert response_body["loss_of_smell"] == predictionList[13]
        assert response_body["itchy_nose"] == predictionList[14]
        assert response_body["itchy_eyes"] == predictionList[15]
        assert response_body["itchy_mouth"] == predictionList[16]
        assert response_body["itchy_inner_ear"] == predictionList[17]
        assert response_body["sneezing"] == predictionList[18]
        assert response_body["pink_eye"] == predictionList[19]
        assert response_body["prediction"] == predictionList[20]
```

ADVANCED

ADVANCED Add-ons are added to the project.

1. Website is hosted on Render [immune.onrender.com] using **gunicorn**
2. Web scraping is employed to extract text to give users a better understanding of the different illness





REFERENCES

- Gov.sg (2020) I am showing respiratory symptoms (like a cough, runny nose or fever), where should I go?, MCI - Gov.SG. Available at: https://www.gov.sg/article/i-am-showing-respiratory-symptoms-where-should-i-go(Accessed: November 14, 2022).
- Teng, A. (2021) Allergy types: Top 12 most common allergies in Singapore, Homage. Available at: https://www.homage.sg/health/allergy-types/ (Accessed: December 2, 2022).
- Conway, W. (2021) Covid, flu, cold symptoms, Kaggle.Available at: https://www.kaggle.com/datasets/walterconway/covid-flu-cold-symptoms(Accessed: November 13, 2022).
- Anon (2021) 800 deaths a year -- and life carries on normally, TWC2. Available at: https://twc2.org.sg/2021/08/31/800-deaths-a-year-and-life-carries-on-normally/ (Accessed: November 26, 2022).