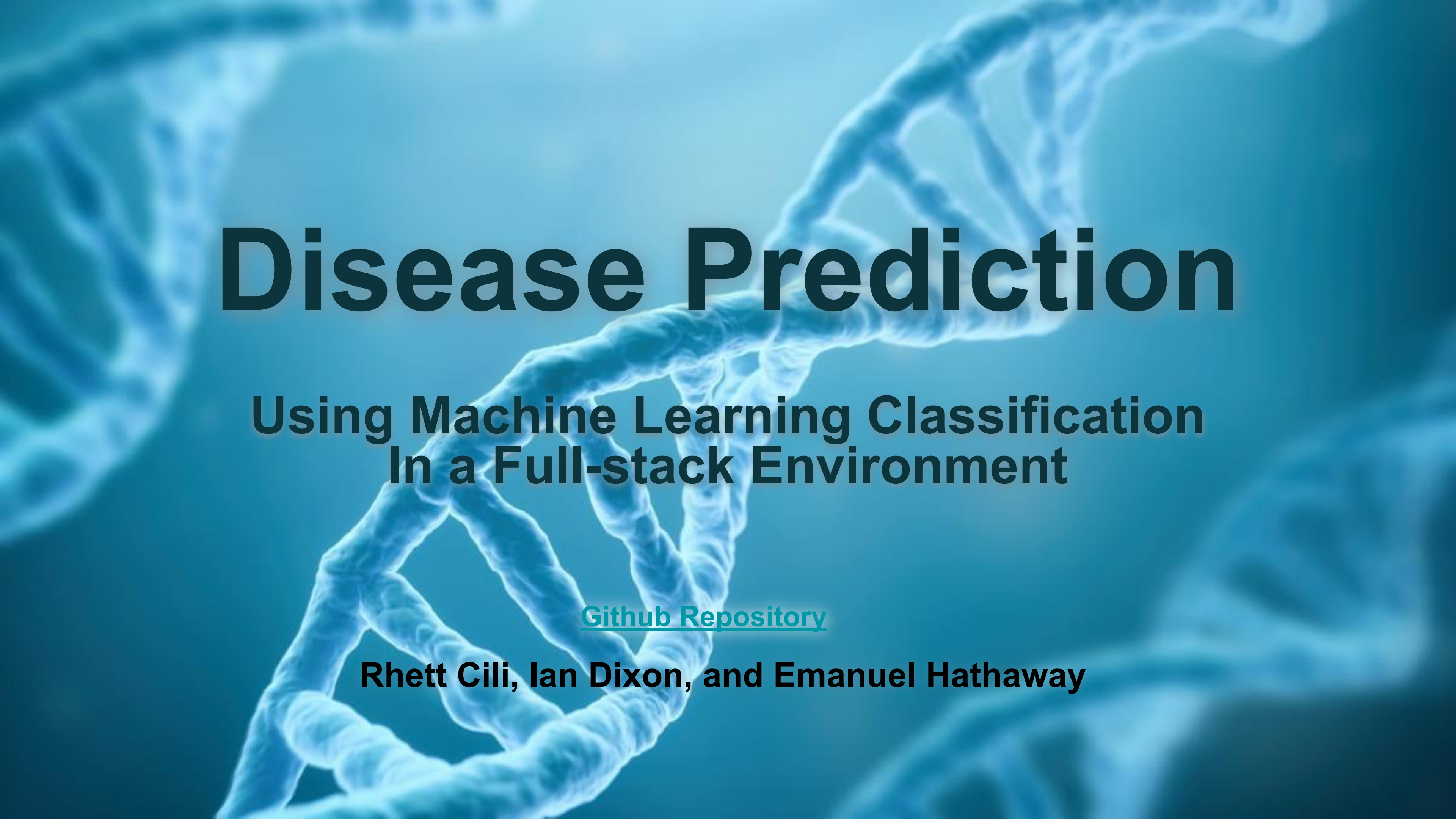


Disease Prediction

Using Machine Learning Classification In a Full-stack Environment

[Github Repository](#)

Rhett Cili, Ian Dixon, and Emanuel Hathaway

Why Disease Prediction?

As a group, we wanted to work on something that we thought could have an impact on people. There are many people around the world suffering from various diseases. While this project will only function as a test, it is a good indication of what machine learning is capable of, and how it may be used in the future of healthcare.

Can we use machine learning classification to predict an illness from experienced symptoms?

Dataset and Exploration

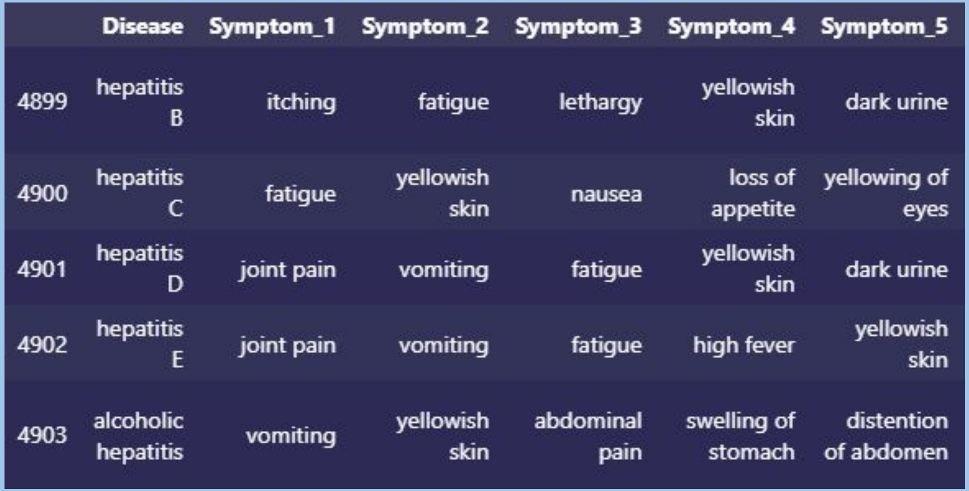
The dataset was originally web-scraped by another team and posted to Kaggle:

- Disease Dataset (shown) consists of 41 diseases and 132 possible symptoms. Each disease has 120 cases.
- Disease Description is a list of the diseases with a brief description of each illness.
- Disease Precaution is a list of precautions to take for each disease.
- Symptom Severity is a list of all symptoms with a weight to indicate severity.

Many replacements were made to the dataset for consistency and clarity, such as correcting typos or retranslating disease and symptom names.

Removing duplicates would drop the sample size for each disease from 120 to 5-10.
Keeping duplicates prevented data overfitting the ML model.

Cleaned in a Jupyter notebook file using Pandas.

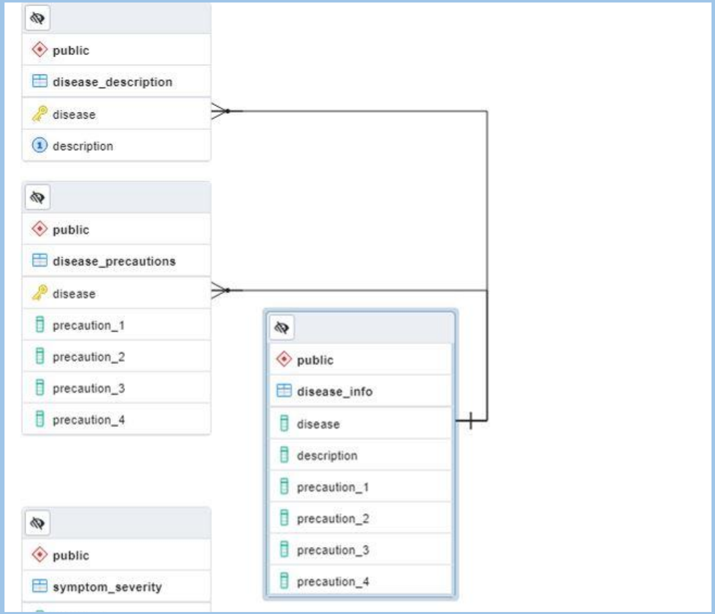


	Disease	Symptom_1	Symptom_2	Symptom_3	Symptom_4	Symptom_5
4899	hepatitis B	itching	fatigue	lethargy	yellowish skin	dark urine
4900	hepatitis C	fatigue	yellowish skin	nausea	loss of appetite	yellowing of eyes
4901	hepatitis D	joint pain	vomiting	fatigue	yellowish skin	dark urine
4902	hepatitis E	joint pain	vomiting	fatigue	high fever	yellowish skin
4903	alcoholic hepatitis	vomiting	yellowish skin	abdominal pain	swelling of stomach	distention of abdomen

Database

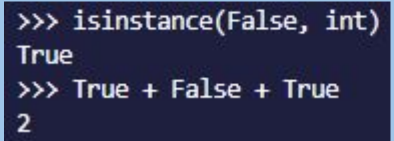
SQL was used to create a relational database with multiple tables for Disease Description, Disease Precautions, and the main dataset.

- The first step in setting up the SQL database with our dataset was to create tables to import the 4 tables.
- With our data imported, we use the "Disease_Descriptions" and "Disease_Precautions" tables to create a new joined table called "Disease_Info" with all information on the diseases.
- This ERD snippet shows the table relationship used to create the disease information table.
- “Disease_Info” will later be queried for disease information after model makes prediction.

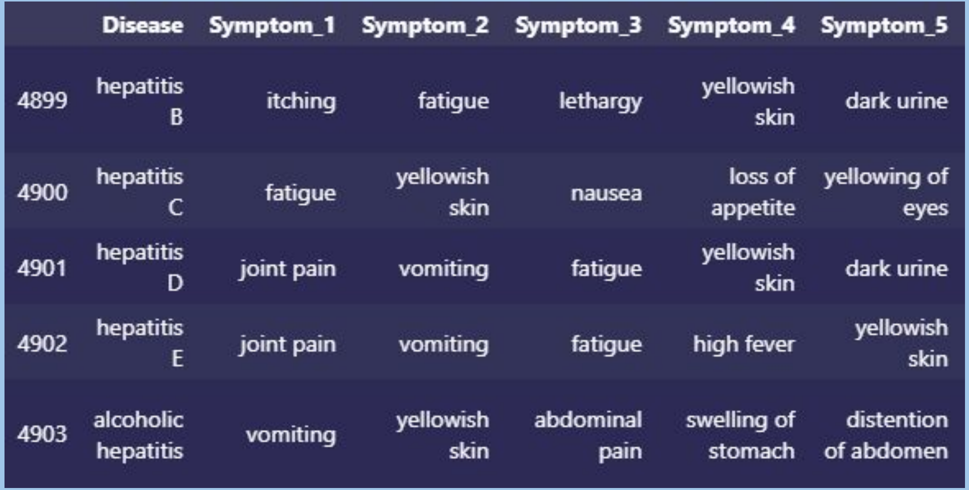


Data Encoding

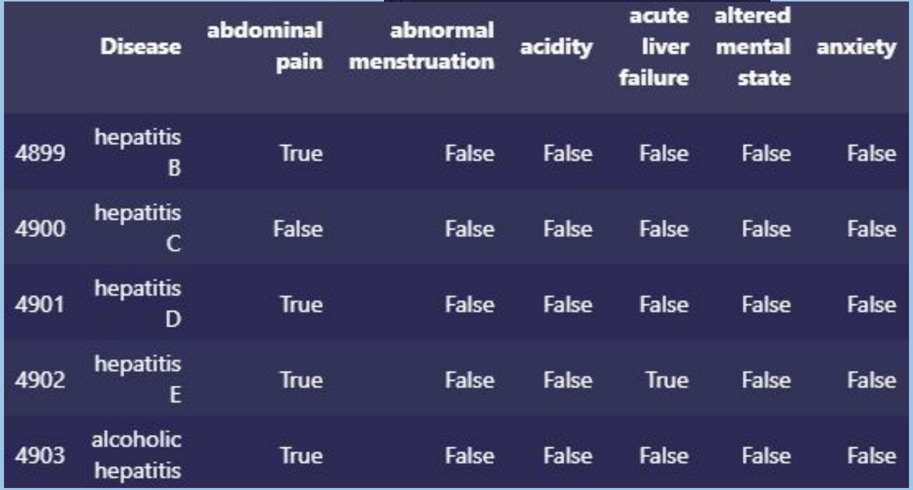
- Dataset format was not suited for one-hot encoding.
- The main dataset of disease symptoms per case was transformed to contain columns for every possible symptom, each containing Boolean values.
- Because Python interprets Booleans as 1 or 0, this format is now encoded and scaled for the machine learning model.
- List comprehension was used to transform DataFrame.



```
>>> isinstance(False, int)
True
>>> True + False + True
2
```



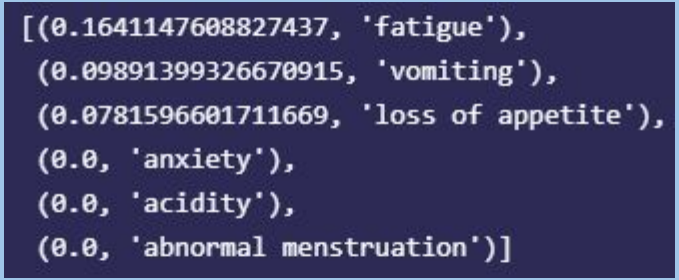
	Disease	Symptom_1	Symptom_2	Symptom_3	Symptom_4	Symptom_5
4899	hepatitis B	itching	fatigue	lethargy	yellowish skin	dark urine
4900	hepatitis C	fatigue	yellowish skin	nausea	loss of appetite	yellowing of eyes
4901	hepatitis D	joint pain	vomiting	fatigue	yellowish skin	dark urine
4902	hepatitis E	joint pain	vomiting	fatigue	high fever	yellowish skin
4903	alcoholic hepatitis	vomiting	yellowish skin	abdominal pain	swelling of stomach	distention of abdomen



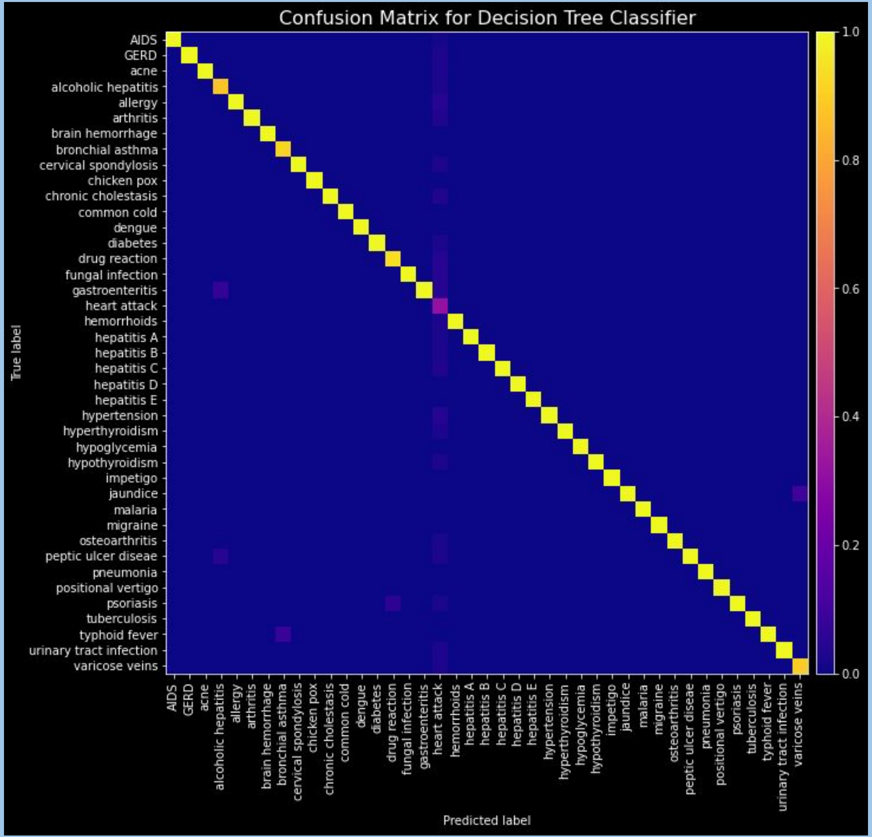
	Disease	abdominal pain	abnormal menstruation	acidity	acute liver failure	altered mental state	anxiety
4899	hepatitis B	True	False	False	False	False	False
4900	hepatitis C	False	False	False	False	False	False
4901	hepatitis D	True	False	False	False	False	False
4902	hepatitis E	True	False	False	True	False	False
4903	alcoholic hepatitis	True	False	False	False	False	False

Machine Learning Model - Benchmark

- Decision Tree Classifier used as a benchmark model
- Many diseases are falsely predicted as heart attack
- 95% accuracy, poor model



```
[(0.1641147608827437, 'fatigue'),
 (0.09891399326670915, 'vomiting'),
 (0.0781596601711669, 'loss of appetite'),
 (0.0, 'anxiety'),
 (0.0, 'acidity'),
 (0.0, 'abnormal menstruation')]
```



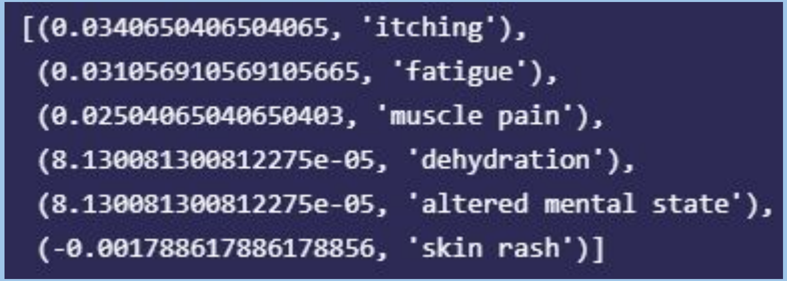
Machine Learning Model - Improved

Support Vector Machines excel at multiclass classification, esp. with clear separation between boundaries in data

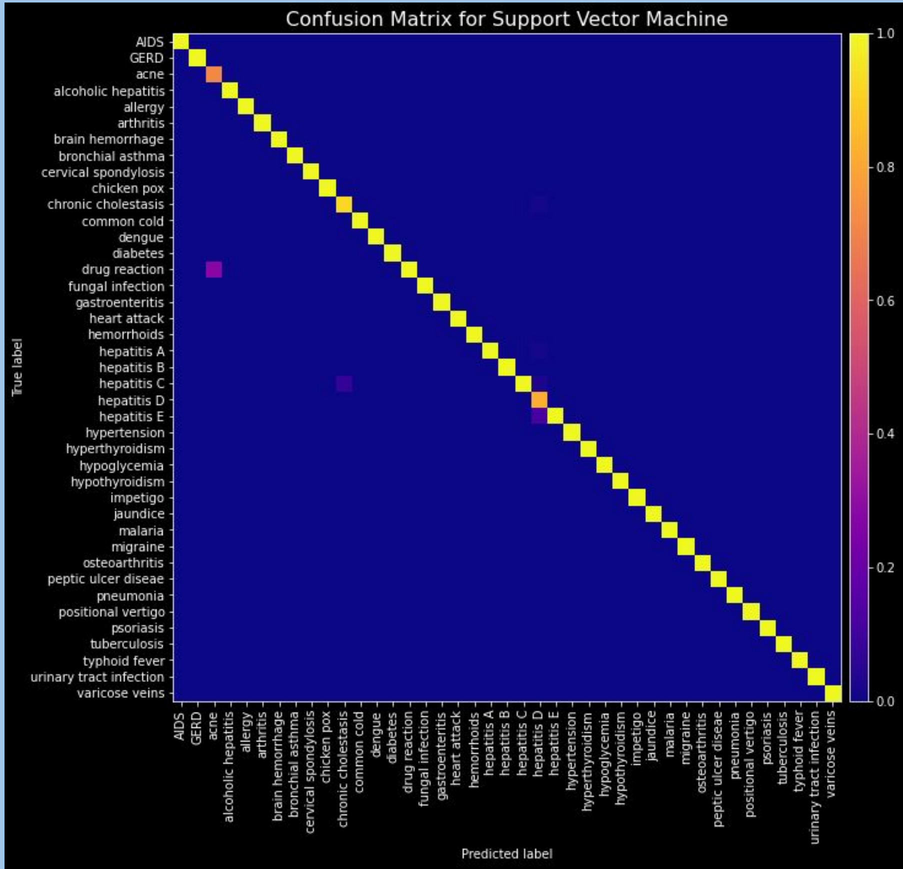
Small effective sample size of dataset mitigated with choice of SVM and 50/50 train/test split

98% accuracy, less worrisome confusion matrix

‘Skin rash’ mildly harmful to model performance

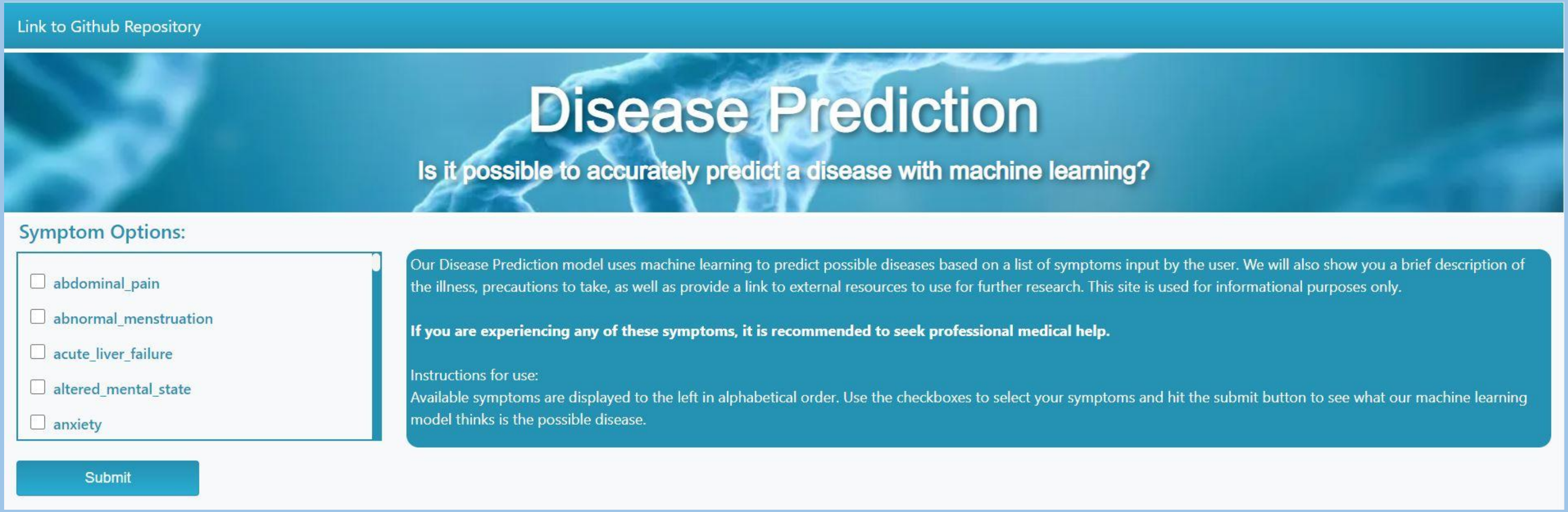


```
[(0.0340650406504065, 'itching'),
 (0.031056910569105665, 'fatigue'),
 (0.02504065040650403, 'muscle pain'),
 (8.130081300812275e-05, 'dehydration'),
 (8.130081300812275e-05, 'altered mental state'),
 (-0.001788617886178856, 'skin rash')]
```



Dashboard Overview and Demo

- Javascript, D3, Python, Flask, HTML/CSS, and SQLAlchemy.
- Allows the user to input symptoms.
- View the model's prediction of their illness, a description, treatment/precautions, and an external link to WebMD for additional info.



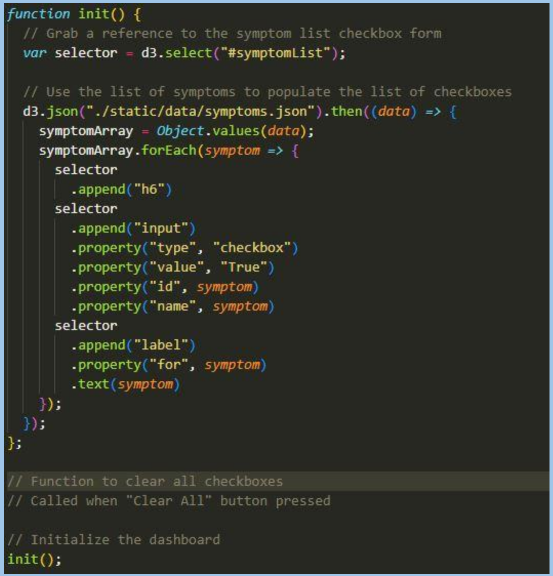
Our Disease Prediction model uses machine learning to predict possible diseases based on a list of symptoms input by the user. We will also show you a brief description of the illness, precautions to take, as well as provide a link to external resources to use for further research. This site is used for informational purposes only.

If you are experiencing any of these symptoms, it is recommended to seek professional medical help.

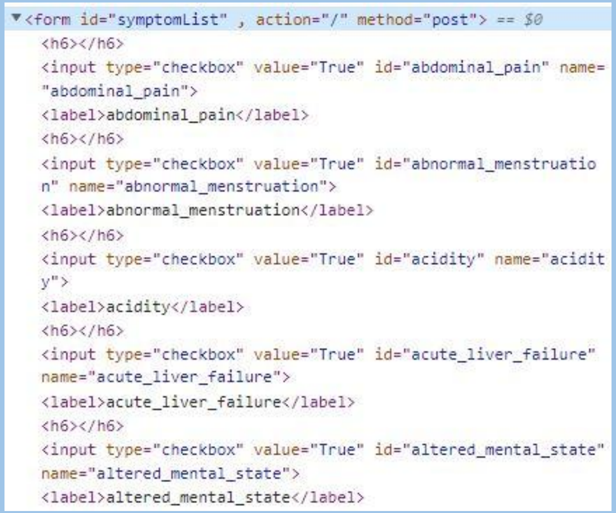
Instructions for use:
Available symptoms are displayed to the left in alphabetical order. Use the checkboxes to select your symptoms and hit the submit button to see what our machine learning model thinks is the possible disease.

Dashboard Frontend - Symptom Options Components

- Uses a <form> tag filled with ‘checkbox’ inputs to collect the data from the user.
- The ‘Symptom Options’ are dynamically populated through a Javascript function, which reads in a JSON file of the symptoms.
- Iterates through each symptom. The function is then called to initialize the index page.
- D3.select used to target the appropriate div class and append the properties for the input tags.



```
function init() {
  // Grab a reference to the symptom list checkbox form
  var selector = d3.select("#symptomList");

  // Use the list of symptoms to populate the list of checkboxes
  d3.json("./static/data/symptoms.json").then((data) => {
    symptomArray = Object.values(data);
    symptomArray.forEach(symptom => {
      selector
        .append("h6")
      selector
        .append("input")
        .property("type", "checkbox")
        .property("value", "True")
        .property("id", symptom)
        .property("name", symptom)
      selector
        .append("label")
        .property("for", symptom)
        .text(symptom)
    });
  });
};

// Function to clear all checkboxes
// Called when "Clear All" button pressed

// Initialize the dashboard
init();
```



```
▼<form id="symptomList" , action="/" method="post"> == $0
    <h6></h6>
    <input type="checkbox" value="True" id="abdominal_pain" name=
    "abdominal_pain">
    <label>abdominal_pain</label>
    <h6></h6>
    <input type="checkbox" value="True" id="abnormal_menstruatio
    n" name="abnormal_menstruation">
    <label>abnormal_menstruation</label>
    <h6></h6>
    <input type="checkbox" value="True" id="acidity" name="acidit
    y">
    <label>acidity</label>
    <h6></h6>
    <input type="checkbox" value="True" id="acute_liver_failure"
    name="acute_liver_failure">
    <label>acute_liver_failure</label>
    <h6></h6>
    <input type="checkbox" value="True" id="altered_mental_state"
    name="altered_mental_state">
    <label>altered_mental_state</label>
```

Dashboard Flask Backend - Data Importing

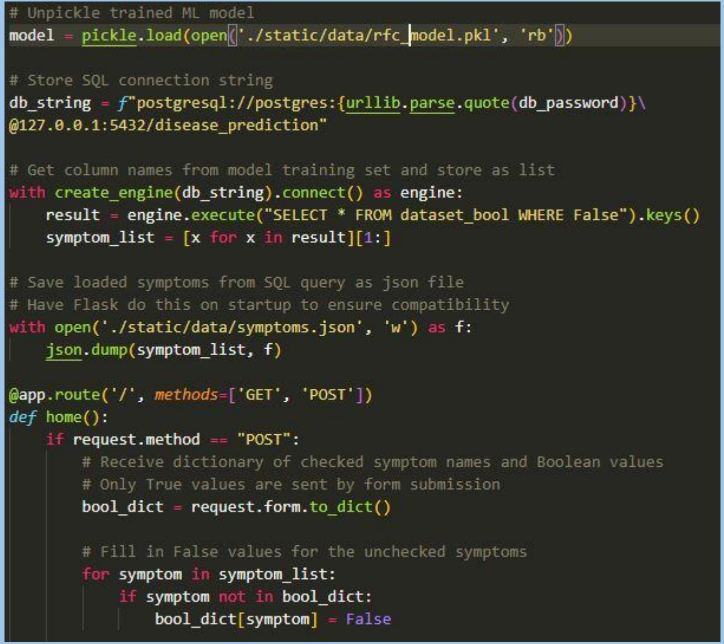
- The trained model was loaded into the app using Pickle.

- SQLAlchemy used to load symptom list and precaution data from Postgres. Symptom list used to generate symptons.json, which is used in our Javascript to populate the symptom options on index.html

- Form data is loaded into the app using Flask’s built in request.form function and converted to a dictionary.

- Request.form function only retrieves the data from ‘checked’ boxes as “True”

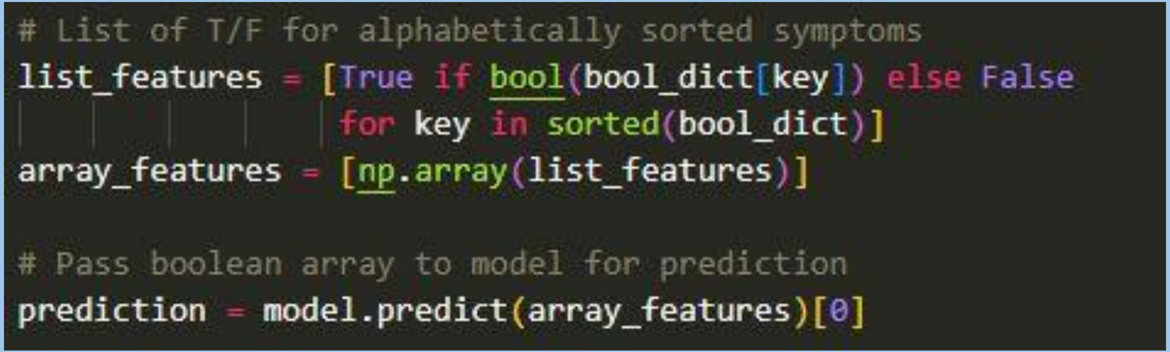
- Used Python to compare the form data with the full list of symptoms and fill in missing values as ‘False’.



```python
# Unpickle trained ML model
model = pickle.load(open('./static/data/rfc_model.pkl', 'rb'))

# Store SQL connection string
db_string = f"postgresql://postgres:{urllib.parse.quote(db_password)}\
@127.0.0.1:5432/disease_prediction"

# Get column names from model training set and store as list
with create_engine(db_string).connect() as engine:
    result = engine.execute("SELECT * FROM dataset_bool WHERE False").keys()
    symptom_list = [x for x in result][1:]

# Save loaded symptoms from SQL query as json file
# Have Flask do this on startup to ensure compatibility
with open('./static/data/symptoms.json', 'w') as f:
    json.dump(symptom_list, f)

@app.route('/', methods=['GET', 'POST'])
def home():
    if request.method == "POST":
        # Receive dictionary of checked symptom names and Boolean values
        # Only True values are sent by form submission
        bool_dict = request.form.to_dict()

        # Fill in False values for the unchecked symptoms
        for symptom in symptom_list:
            if symptom not in bool_dict:
                bool_dict[symptom] = False
```

Dashboard Flask Backend - ML Data Loading

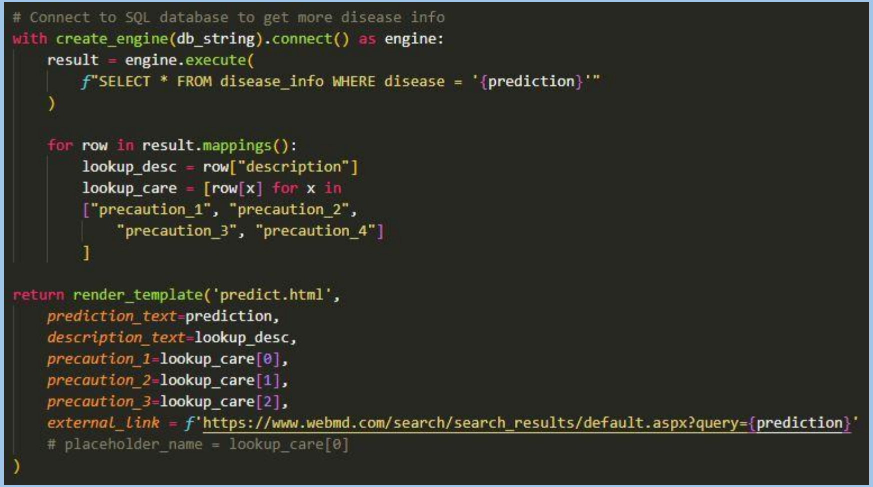
With the data imported into our app, we used Python to transform the data into the proper format for the ML model to accept and saved the output as {prediction} variable.



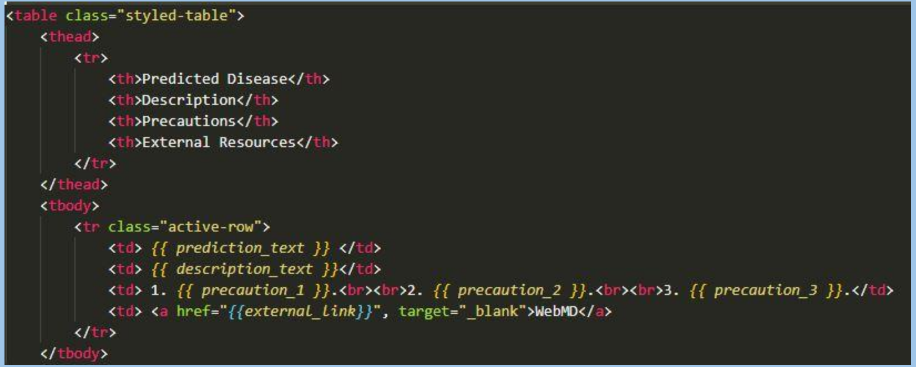
```python
# List of T/F for alphabetically sorted symptoms
list_features = [True if bool(bool_dict[key]) else False
                 for key in sorted(bool_dict)]
array_features = [np.array(list_features)]

# Pass boolean array to model for prediction
prediction = model.predict(array_features)[0]
```

Dashboard Flask Backend - Table Population

- SQLAlchemy used to pull in our corresponding disease data and assign variables.
- All variables are then passed into Flask’s render_template() function, which passes the data back to the front end to populate our data table on predict.html



```
# Connect to SQL database to get more disease info
with create_engine(db_string).connect() as engine:
    result = engine.execute(
        f"SELECT * FROM disease_info WHERE disease = '{prediction}'"
    )

    for row in result.mappings():
        lookup_desc = row["description"]
        lookup_care = [row[x] for x in
        ["precaution_1", "precaution_2",
            "precaution_3", "precaution_4"]
        ]

return render_template('predict.html',
    prediction_text=prediction,
    description_text=lookup_desc,
    precaution_1=lookup_care[0],
    precaution_2=lookup_care[1],
    precaution_3=lookup_care[2],
    external_link = f'https://www.webmd.com/search/search_results/default.aspx?query={prediction}'
    # placeholder_name = lookup_care[0]
)
```



```
<table class="styled-table">
    <thead>
        <tr>
            <th>Predicted Disease</th>
            <th>Description</th>
            <th>Precautions</th>
            <th>External Resources</th>
        </tr>
    </thead>
    <tbody>
        <tr class="active-row">
            <td> {{ prediction_text }} </td>
            <td> {{ description_text }}</td>
            <td> 1. {{ precaution_1 }}.<br><br>2. {{ precaution_2 }}.<br><br>3. {{ precaution_3 }}.</td>
            <td> <a href="{{external_link}}", target="_blank">WebMD</a>
        </tr>
    </tbody>
```

Future Improvements

- Higher quality dataset
- Using confusion matrix to suggest differential diagnosis
- Sort symptom options into categories for ease of user access
- Public hosting of webpage and database with Heroku and Amazon RDS

Conclusion

Can we use machine learning classification to predict an illness from experienced symptoms?

YES!

While our model had some bugs due to the size and quality of the dataset, machine learning does prove to be a useful tool in the future of healthcare.