Random Forest Classification of Rheumatic and Autoimmune Diseases

Jesse Byers

College of Information Technology, Western Governors University

Dr. Daniel Smith (Instructor)

September 11, 2025

Random Forest Classification of Rheumatic and Autoimmune Diseases

Research Question

The goal of this study is to create a predictive model to support early diagnosis of difficult-to-diagnose rheumatic and autoimmune diseases. Rheumatic and autoimmune diseases can be very challenging to diagnose and treat because many diseases have overlapping, nonspecific subjective symptoms, such as fatigue, joint pain, stiffness, and brain fog. In addition, many patients who report these types of symptoms are dismissed and told the symptoms may be due to getting older, being overweight, or being depressed. As a result, many individuals have long delays in getting a proper diagnosis and starting treatment for their condition, especially women and patients from underserved communities (Myers, 2025). In addition, there is data to suggest that autoimmune diseases are becoming more prevalent and can be very costly to treat (Miller, 2023).

Maching Learning techniques can be explored to determine whether diagnoses can be made based on patterns in objective signs, such as blood test results and genetic markers (Mahdi, Jahani, & Abd, 2025). If a predictive model is successful, then patients could receive faster diagnoses, with less risk of having subjective symptoms dismissed. This could result in earlier treatment and better patient outcomes, with less overall health care costs.

The validity and accuracy of a predictive model for diagnosing autoimmune diseases is dependent on choosing the right objective markers for analysis. Thus, the primary research question is: To what extent do blood test result variables (including ESR, CRP, RF, Anti-CCP,

HLA-B27, ANA, Anti-Ro, Anti-La, Anti-dsDNA, Anti-Sm, C3, and C4) predict a specific autoimmune disease diagnosis? The null hypothesis is that the blood test result variables alone can not accurately predict a disease. The alternate hypothesis is that the blood test result variables statistically significantly predict a disease.

Data Collection

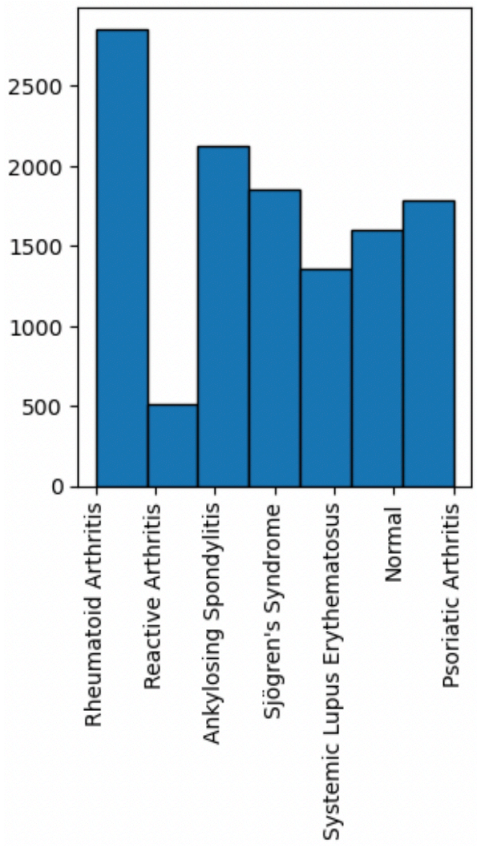
The *Diagnosis of rheumatic and autoimmune diseases dataset* will be used for this analysis. The dataset includes 12,085 rows of patient data from a single hospital in Iraq, and includes age, gender, and 12 features from blood test results (a mix of categorical and numeric continuous data types). The blood test results are sourced from three independent laboratories.

Feature	Data Type
Age	Numeric (continuous)
Sex	Categorical
ESR	Numeric (continuous)
CRP	Numeric (continuous)
RF	Numeric (continuous)
Anti-CCP	Numeric (continuous)
HLA-B27	Categorical
ANA	Categorical
Anti-Ro	Categorical
Anti-La	Categorical
Anti-dsDNA	Categorical
Anti-Sm	Categorical
C3	Numeric (continuous)
C4	Numeric (continuous)

The patients fall into seven disease classes, as summarized below (Mahdi, Jahani, & Abd, 2025).

Disease Label	Number of Samples
Rheumatoid Arthritis	2,848
Reactive Arthritis	516
Ankylosing Spondylitis	2,127
Sjogren's Syndrome	1,852
Systemic Lupus Erythematosus	1,355
Psoriatic Arthritis	1,783
Normal (no disease)	1,604

This data is free and publicly available at the Harvard Dataverse website (Abd, Mahdi, & Jahani, 2025). The data use is governed by the Creative Commons 1.0 Universal License, which allows use without explicit permission (Creative Commons, 2019). The data has already been gathered and is available for download through a weblink as an Excel file (Abd, Mahdi, & Jahani, 2025). Overall, the data is easy to access and use, and did not present any challenges at the data collection stage. See Appendix 1 for the Excel data file.



The primary advantage of using this dataset for analysis is in its size and scope. With over 12,000 samples across 7 disease classes, there is a sufficient number of samples representing each disease class. However, despite the overall size, the classes are unbalanced, with only approximately 500 samples representing the smallest class (Reactive Arthritis) and almost 3,000 representing the largest class (Rheumatoid Arthritis). Because the samples were collected from a hospital over a three-year period, they should represent the general incidence of each disease within the population. The unbalanced nature of the classes, although it likely reflects real-world patterns, could potentially impact the training of a predictive model, as the model could learn to over-predict the over-represented classes.

The primary disadvantage of using this dataset relates to the sparsity of blood test result values. While the dataset includes a robust set of 12 variables relevant to autoimmune and rheumatic disease diagnosis, the dataset as a whole includes a large number of missing values. Out of 12,085 patients, only 452 patients, or less than 5% of patients, have a full set of blood test results with no missing values.

Feature	Number of Samples with Missing Values	Percent of Samples with Missing Values
Age	0	0%
Sex	0	0%
ESR	1088	9%
CRP	2417	20%
RF	1329	11%
Anti-CCP	3263	27%
HLA-B27	1934	16%
ANA	3746	31%
Anti-Ro	2900	24%
Anti-La	3021	25%
Anti-dsDNA	4713	39%
Anti-Sm	5197	43%
C3	1692	14%
C4	2054	17%
Disease	0	0%

While missing values can be challenging to address, the prevalence of these missing values in the dataset is realistic. Specific blood tests are ordered for individual patients based on factors such as symptoms, risk factors, etc., so it would be unreasonable to expect a real medical dataset to have a full set of values for each patient. However, the volume of missing values and how they are addressed could have a large impact on the results of a predictive model trained on this data. The challenges and strategies for addressing these missing values will be discussed in the next section.

Data Extraction and Preparation

The data required preprocessing to detect and address missing values in the dataset, identify trends and relationships, encode categorical variables, and split the data into training, validation, and testing sets.

Detect and Address Missing Values

As noted above, less than 5% of the samples included a full set of blood test result values. This is a very common problem in medical datasets that rely on clinical data. According to Austin et. al., "Data can be missing for several reasons, including: (i) patient refusal to respond to specific questions (eg, patient does not report data on income); (ii) loss of patient to follow-up; (iii) investigator or mechanical error (eg, sphygmomanometer failure); and (iv) physicians not ordering certain investigations for some patients (eg, cholesterol test not ordered for some patients)." (Austin et. al., 2020). In the case of this dataset, in relation to autoimmune and rheumatic disorders, the likely cause of most missing values is that the physician did not order certain tests for certain patients, based on the results of other tests, as well as their clinical presentation of symptoms. This type of missingness can be characterized as Missing Not At Random (MNAR) since the missing values are likely dependent on the physician's clinical judgment related to factors that are not recorded in the dataset itself (Austin et. al., 2020).

With the type of missingness identified as MNAR, four different approaches for addressing the missing values were explored:

Option 1: Delete all rows with missing values

```
In [4]: # option 1 - drop all rows with any missing values
        df_delete_missing = df_raw.dropna(axis=0, how="any")
        display(df_delete_missing)
            # only 452 rows remaining with all data points
```

	Age	Gender	ESR	CRP	RF	Anti-CCP	HLA-B27	ANA	Anti-Ro	Anti-La	Anti-dsDNA	Anti-Sm	C3	C4	Disease
10	24	Male	31.0	13.7	31.6	27.7	Positive	Negative	Positive	Positive	Negative	Negative	170.0	57.0	Rheumatoid Arthritis
57	35	Male	43.0	25.7	30.5	39.9	Positive	Negative	Positive	Positive	Positive	Positive	114.0	40.0	Rheumatoid Arthritis
74	42	Female	32.0	17.0	39.8	21.3	Positive	Negative	Positive	Negative	Negative	Negative	130.0	29.0	Rheumatoid Arthritis
92	38	Male	30.0	13.2	37.3	33.5	Positive	Negative	Negative	Positive	Positive	Negative	160.0	47.0	Rheumatoid Arthritis
126	48	Female	46.0	11.6	39.8	39.1	Positive	Positive	Positive	Positive	Positive	Positive	169.0	46.0	Rheumatoid Arthritis
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
11904	28	Female	35.0	16.2	9.0	17.8	Negative	Negative	Positive	Positive	Positive	Positive	129.0	17.0	Psoriatic Arthritis
11913	31	Male	34.0	21.8	2.9	10.7	Positive	Negative	Negative	Negative	Negative	Negative	100.0	37.0	Psoriatic Arthritis
11932	51	Female	43.0	27.3	0.1	12.5	Negative	Negative	Positive	Positive	Negative	Negative	120.0	55.0	Psoriatic Arthritis
11977	76	Female	41.0	17.1	8.4	8.4	Negative	Negative	Negative	Negative	Positive	Positive	146.0	52.0	Psoriatic Arthritis
12032	67	Male	32.0	18.7	1.8	13.8	Positive	Negative	Negative	Negative	Positive	Negative	162.0	55.0	Psoriatic Arthritis

452 rows × 15 columns

This approach results in a small data set with only 452 samples, which is too small for training a classification model. This represents less than 5% of the original patients.

Option 2: Impute missing values with the midpoint of the normal reference range

Since the missing values are likely dependent on the physician's clinical judgment that the test was not needed, an assumption could be made that the untested values are likely within the normal range for each test. Research was conducted to identify the LabCorp normal reference range for each test, as well as the midpoint of the reference range. Missing values were imputed with these midpoints on a test-by-test basis.

In [5]:
```
# option 2 – impute all missing values with midpoint of normal reference range from the test
    # assumption that the value is normal if doctors did not choose to order the test
    # using reference ranges from Labcorp (see links to URL for each test)

df_normals = pd.DataFrame([
    {"test": "ESR", "min": 0, "max": 40, "normal": 20.0},  # https://www.labcorp.com/tests/005215/sedimentation-rate-modified-westergren
    {"test": "CRP", "min": 0, "max": 10, "normal": 5.0},   # https://www.labcorp.com/tests/006627/c-reactive-protein-crp-quantitative
    {"test": "RF", "min": 0, "max": 14, "normal": 7.0},    # https://www.labcorp.com/tests/006502/rheumatoid-factor-rf
    {"test": "Anti-CCP", "min": 0, "max": 20, "normal": 10.0},  # https://www.labcorp.com/tests/164914/anti-ccp-cyclic-citrullinated-peptide-antibodies-igg-and-iga-elisa
    {"test": "HLA-B27", "min": "Negative", "max": "Negative", "normal": "Negative"},  # https://www.labcorp.com/tests/006924/hla-b-27-disease-association
    {"test": "ANA", "min": "Negative", "max": "Negative", "normal": "Negative"},  # https://www.labcorp.com/tests/164947/ana-by-ifa-reflex-to-titer-and-pattern
    {"test": "Anti-Ro", "min": "Negative", "max": "Negative", "normal": "Negative"},  # https://www.labcorp.com/tests/520010/anti-ro-ss-a-ab-rdl
    {"test": "Anti-La", "min": "Negative", "max": "Negative", "normal": "Negative"},  # https://www.labcorp.com/tests/520320/anti-la-ss-b-ab-rdl
    {"test": "Anti-dsDNA", "min": "Negative", "max": "Negative", "normal": "Negative"}, # https://www.labcorp.com/tests/096339/anti-dsdna-double-stranded-antibodies
    {"test": "Anti-Sm", "min": "Negative", "max": "Negative", "normal": "Negative"},  # https://www.labcorp.com/tests/520013/anti-sm-ab-rdl
    {"test": "C3", "min": 82, "max": 167, "normal": 125.0},  # https://www.labcorp.com/tests/006452/complement-c3
    {"test": "C4", "min": 12, "max": 38, "normal": 25.0}  # https://www.labcorp.com/tests/001834/complement-c4
    ])

print(df_normals)

# impute missing values for each column with df_normals normal value
pd.set_option('future.no_silent_downcasting', True)
df_imputed_normals = pd.DataFrame()
df_imputed_normals["Age"] = df_raw["Age"]
df_imputed_normals["Gender"] = df_raw["Gender"]

num_cols = ["ESR", "CRP", "RF", "Anti-CCP",  "C3", "C4"]
cat_cols = ["HLA-B27", "ANA", "Anti-Ro", "Anti-La", "Anti-dsDNA", "Anti-Sm"]

for col in num_cols:
    test_row = df_normals.loc[df_normals["test"] == col, "normal"]
    val = test_row.iloc[0]
    df_imputed_normals[col] = df_raw[col].astype(float).fillna(val)


for col in cat_cols:
    test_row = df_normals.loc[df_normals["test"] == col, "normal"]
    val = test_row.iloc[0]
    df_imputed_normals[col] = df_raw[col].astype(object).fillna(val)

df_imputed_normals["Disease"] = df_raw["Disease"]

display(df_imputed_normals)
```

```
          test       min       max    normal
0          ESR         0        40      20.0
1          CRP         0        10       5.0
2           RF         0        14       7.0
3     Anti-CCP         0        20      10.0
4      HLA-B27  Negative  Negative  Negative
5          ANA  Negative  Negative  Negative
6      Anti-Ro  Negative  Negative  Negative
7      Anti-La  Negative  Negative  Negative
8   Anti-dsDNA  Negative  Negative  Negative
9      Anti-Sm  Negative  Negative  Negative
10          C3        82       167     125.0
11          C4        12        38      25.0
```

	Age	Gender	ESR	CRP	RF	Anti-CCP	C3	C4	HLA-B27	ANA	Anti-Ro	Anti-La	Anti-dsDNA	Anti-Sm	Disease
0	70	Male	39.0	18.6	34.2	29.9	125.0	27.0	Positive	Negative	Positive	Negative	Positive	Positive	Rheumatoid Arthritis
1	39	Female	26.0	21.7	35.5	28.9	100.0	66.0	Negative	Negative	Positive	Negative	Positive	Negative	Rheumatoid Arthritis
2	36	Female	41.0	15.6	21.3	21.3	158.0	12.0	Negative	Negative	Negative	Positive	Negative	Negative	Rheumatoid Arthritis
3	35	Male	43.0	23.4	26.0	39.0	119.0	41.0	Negative	Negative	Positive	Positive	Negative	Negative	Rheumatoid Arthritis
4	37	Female	30.0	5.0	38.1	30.8	144.0	49.0	Positive	Negative	Positive	Negative	Positive	Negative	Rheumatoid Arthritis
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
12080	32	Male	36.0	17.0	14.5	16.1	125.0	32.0	Negative	Positive	Negative	Positive	Positive	Negative	Psoriatic Arthritis
12081	36	Male	43.0	5.0	17.7	13.5	125.0	41.0	Negative	Negative	Negative	Positive	Negative	Negative	Psoriatic Arthritis
12082	20	Female	31.0	28.8	4.8	5.8	125.0	25.0	Negative	Positive	Positive	Negative	Negative	Negative	Psoriatic Arthritis
12083	33	Female	36.0	5.0	7.0	9.5	96.0	52.0	Positive	Positive	Positive	Positive	Positive	Negative	Psoriatic Arthritis
12084	48	Female	32.0	26.9	7.2	13.6	178.0	25.0	Positive	Negative	Negative	Negative	Negative	Negative	Psoriatic Arthritis

12085 rows × 15 columns

This approach retains the full sample size in the dataset, but could be problematic based on the inaccuracies in the imputed values. Each laboratory may have its own reference ranges for normal for each test, and these ranges can differ from laboratory to laboratory. Thus, the

midpoints identified from the LabCorp laboratory may not be an accurate representation of the normal reference range used by the three laboratories in Iraq, which are not identified by name. In addition, the output dataset retains no information on which values are real values and which are imputed.

Option 3: Impute missing values with the mean of similar patients

This approach to imputation is based on the assumption that the missing lab values for a particular patient would be similar to others in the same disease cohort. Patients were grouped by disease, and the mean value for each lab was calculated for each group. These values were used to impute the missing values for each test.

```
In [6]: # option 3: impute missing values with mean of those who do / do not have specific disease
            # calculate mean for each numerical test value for those with disease vs those without disease
            # calculate mode (most frequent value) for each categorical test value for those with disease vs those without disease
            # the aggreagate functions excludes missing values when calculating

        num_means = df_raw.groupby("Disease")[num_cols].mean()

        cat_modes = df_raw.groupby("Disease")[cat_cols].agg(pd.Series.mode)

        df_means = pd.concat([num_means, cat_modes], axis=1)

        print(df_means)

        # impute missing values with appropriate values from df_means
        df_imputed_means = df_raw.copy()

        for test in df_means.columns:
            df_imputed_means[test] = df_raw[test].fillna(
                df_raw["Disease"].map(df_means[test])
            )

        df_imputed_means["Disease"] = df_raw["Disease"]

        display(df_imputed_means)
```

```
                                ESR        CRP         RF   Anti-CCP  \
Disease
Ankylosing Spondylitis       32.348140  21.583884  19.997699  19.702078
Normal                        8.125860   1.573401  10.196017   9.759237
Psoriatic Arthritis          40.165325  22.998198   9.951151   9.928812
Reactive Arthritis           24.921277  15.454745  20.078161  19.922626
Rheumatoid Arthritis         35.087409  20.619524  30.369862  30.575257
Sjögren's Syndrome            8.194920   1.512170  20.052699  20.145808
Systemic Lupus Erythematosus  8.334416   1.546330  19.878607  20.233728

                                    C3         C4   HLA-B27       ANA  \
Disease
Ankylosing Spondylitis       140.879757  42.777591  Positive  Negative
Normal                       140.281546  42.020942  Positive  Positive
Psoriatic Arthritis          134.223082  41.419529  Positive  Negative
Reactive Arthritis           140.889391  42.536364  Positive  Positive
Rheumatoid Arthritis         141.927265  41.967066  Negative  Positive
Sjögren's Syndrome           140.608176  42.505906  Negative  Positive
Systemic Lupus Erythematosus  68.521478   6.512478  Positive  Positive

                              Anti-Ro   Anti-La Anti-dsDNA   Anti-Sm
Disease
Ankylosing Spondylitis       Positive  Positive   Negative  Negative
Normal                       Negative  Negative   Negative  Negative
Psoriatic Arthritis          Positive  Positive   Negative  Positive
Reactive Arthritis           Positive  Negative   Positive  Positive
Rheumatoid Arthritis         Negative  Positive   Negative  Negative
Sjögren's Syndrome           Positive  Positive   Positive  Negative
Systemic Lupus Erythematosus Negative  Positive   Positive  Positive
```

	Age	Gender	ESR	CRP	RF	Anti-CCP	HLA-B27	ANA	Anti-Ro	Anti-La	Anti-dsDNA	Anti-Sm	C3	C4	Disease
0	70	Male	39.0	18.600000	34.200000	29.9	Positive	Negative	Positive	Negative	Positive	Positive	141.927265	27.000000	Rheumatoid Arthritis
1	39	Female	26.0	21.700000	35.500000	28.9	Negative	Positive	Positive	Positive	Positive	Negative	100.000000	66.000000	Rheumatoid Arthritis
2	36	Female	41.0	15.600000	21.300000	21.3	Negative	Negative	Negative	Positive	Negative	Negative	158.000000	12.000000	Rheumatoid Arthritis
3	35	Male	43.0	23.400000	26.000000	39.0	Negative	Positive	Positive	Positive	Negative	Negative	119.000000	41.000000	Rheumatoid Arthritis
4	37	Female	30.0	20.619524	38.100000	30.8	Positive	Negative	Positive	Negative	Positive	Negative	144.000000	49.000000	Rheumatoid Arthritis
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
12080	32	Male	36.0	17.000000	14.500000	16.1	Negative	Positive	Negative	Positive	Positive	Positive	134.223082	32.000000	Psoriatic Arthritis
12081	36	Male	43.0	22.998198	17.700000	13.5	Negative	Negative	Negative	Positive	Negative	Negative	134.223082	41.000000	Psoriatic Arthritis
12082	20	Female	31.0	28.800000	4.800000	5.8	Negative	Positive	Positive	Negative	Negative	Negative	134.223082	41.419529	Psoriatic Arthritis
12083	33	Female	36.0	22.998198	9.951151	9.5	Positive	Positive	Positive	Positive	Positive	Positive	96.000000	52.000000	Psoriatic Arthritis
12084	48	Female	32.0	26.900000	7.200000	13.6	Positive	Negative	Positive	Negative	Negative	Positive	178.000000	41.419529	Psoriatic Arthritis

12085 rows × 15 columns

While this approach could be used to impute values to aid in exploratory data analysis, it is an inappropriate strategy for predictive analysis. By grouping patients by disease class in order to calculate means, there is a risk of data leakage because the imputed lab values are influenced by the target disease label. As a result, this approach is invalid for this predictive analysis.

Option 4: Use SimpleImputer to impute mean values and add an indicator flag for missingness

To avoid the risk of target leakage, mean values were calculated for each test from the entire dataset. Scikit-Learn's SimpleImputer was used to replace the missing values with the mean (for numerical tests) or the most frequent value (for categorical tests). In addition, the SimpleImputer added a binary column for each column that had missing values, which indicates whether the row includes a real value or an imputed value.

```
In [7]: # option 4: Use SimpleImputer to impute mean/most frequent values and add indicator flag for missingness

df_simple_imputer = df_raw.copy()

num_transformer = SimpleImputer(missing_values=np.nan, strategy="median", add_indicator=True)
num_imputed = num_transformer.fit_transform(df_simple_imputer[num_cols])
num_out_cols = num_cols + [f"{col}_missing" for col in num_cols]
df_num = pd.DataFrame(num_imputed, columns=num_out_cols, index=df_simple_imputer.index)

cat_transformer = SimpleImputer(strategy="most_frequent", add_indicator=True)
cat_imputed = cat_transformer.fit_transform(df_simple_imputer[cat_cols])
cat_out_cols = cat_cols + [f"{col}_missing" for col in cat_cols]
df_cat = pd.DataFrame(cat_imputed, columns=cat_out_cols, index=df_simple_imputer.index)

df_simple_imputer = df_simple_imputer.drop(columns=num_cols + cat_cols)
df_simple_imputer = pd.concat([df_simple_imputer, df_num, df_cat], axis=1)

display(df_simple_imputer)
```

	Age	Gender	Disease	ESR	CRP	RF	Anti-CCP	C3	C4	ESR_missing	...	Anti-Ro	Anti-La	Anti-dsDNA	Anti-Sm	HLA-B27_missing	ANA_missing	Anti-Ro_missing
0	70	Male	Rheumatoid Arthritis	39.0	18.6	34.2	29.9	133.0	27.0	0.0	...	Positive	Negative	Positive	Positive	False	False	False
1	39	Female	Rheumatoid Arthritis	26.0	21.7	35.5	28.9	100.0	66.0	0.0	...	Positive	Positive	Positive	Positive	False	True	False
2	36	Female	Rheumatoid Arthritis	41.0	15.6	21.3	21.3	158.0	12.0	0.0	...	Positive	Positive	Negative	Positive	False	False	True
3	35	Male	Rheumatoid Arthritis	43.0	23.4	26.0	39.0	119.0	41.0	0.0	...	Positive	Positive	Positive	Positive	True	True	False
4	37	Female	Rheumatoid Arthritis	30.0	15.6	38.1	30.8	144.0	49.0	0.0	...	Positive	Negative	Positive	Negative	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
12080	32	Male	Psoriatic Arthritis	36.0	17.0	14.5	16.1	133.0	32.0	0.0	...	Negative	Positive	Positive	Positive	False	False	False
12081	36	Male	Psoriatic Arthritis	43.0	15.6	17.7	13.5	133.0	41.0	0.0	...	Negative	Positive	Positive	Negative	False	False	False
12082	20	Female	Psoriatic Arthritis	31.0	28.8	4.8	5.8	133.0	38.0	0.0	...	Positive	Negative	Negative	Negative	False	False	False
12083	33	Female	Psoriatic Arthritis	36.0	15.6	19.2	9.5	96.0	52.0	0.0	...	Positive	Positive	Positive	Positive	False	False	False
12084	48	Female	Psoriatic Arthritis	32.0	26.9	7.2	13.6	178.0	38.0	0.0	...	Positive	Negative	Positive	Positive	False	True	True

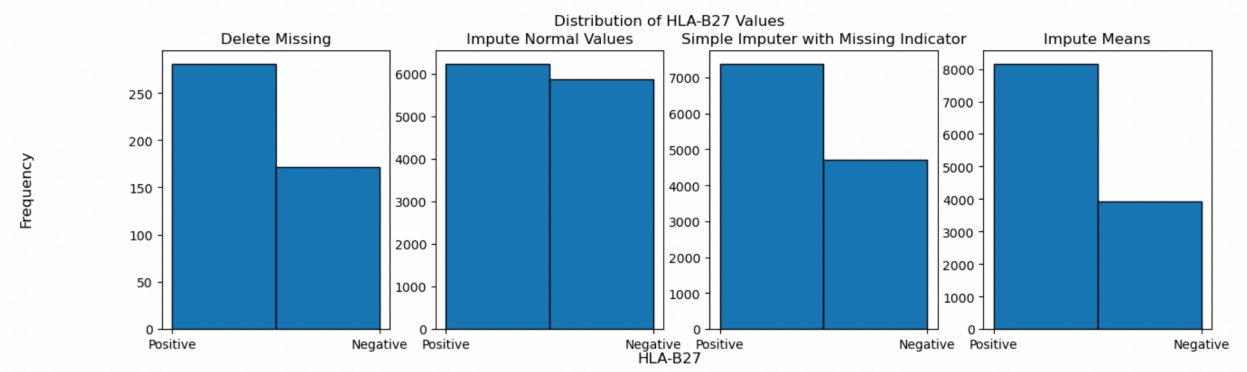
12085 rows × 27 columns

This approach has the benefit of retaining the information on which tests were not ordered for each patient, which could be a proxy representing the hidden clinical judgment of the physician. The missing flags could have predictive value in the final model.

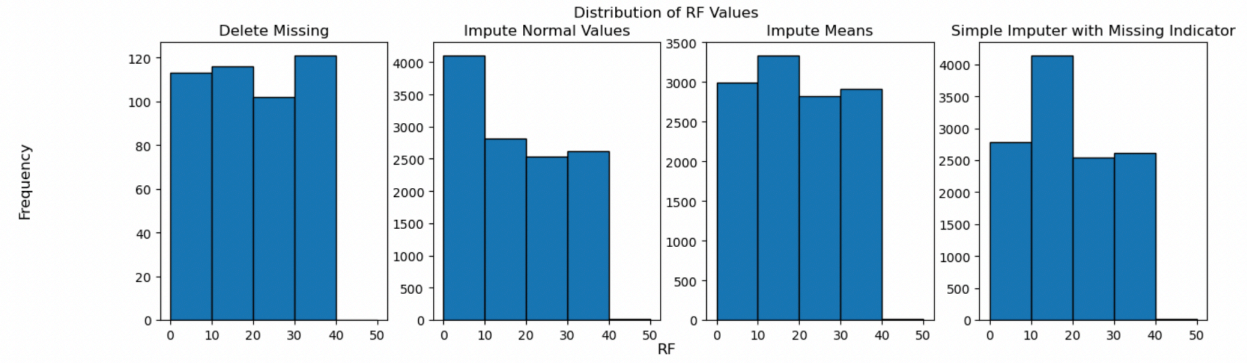
Evaluating the Four Options

As noted above, two of the options were immediately eliminated from consideration. Option 1 (delete all rows with missing values) would lead to an unreasonably small final data set. Option 3 (impute missing values with the mean of similar patients) would introduce target leakage and would likely lead to bias and overfitting in a predictive model.

A series of histograms was generated to evaluate the distribution of values for each variable using the four options. Overall, the four approaches to addressing missing values had differing impacts on the distribution of lab values for each test. For example, the charts below show how the distributions of positive and negative values for the HLA-B27 test (a categorical variable) change based on the approach to addressing missing values. In this case, replacing all missing values with the normal value (negative) makes the number of positive and negative samples almost equal, while all other approaches retain the majority of samples as positive.



For a numerical variable example, the RF test charts show that the imputation of normal values skews the distribution towards the lowest (normal) values, while the Simple Imputer option shifts the distribution more towards the right.



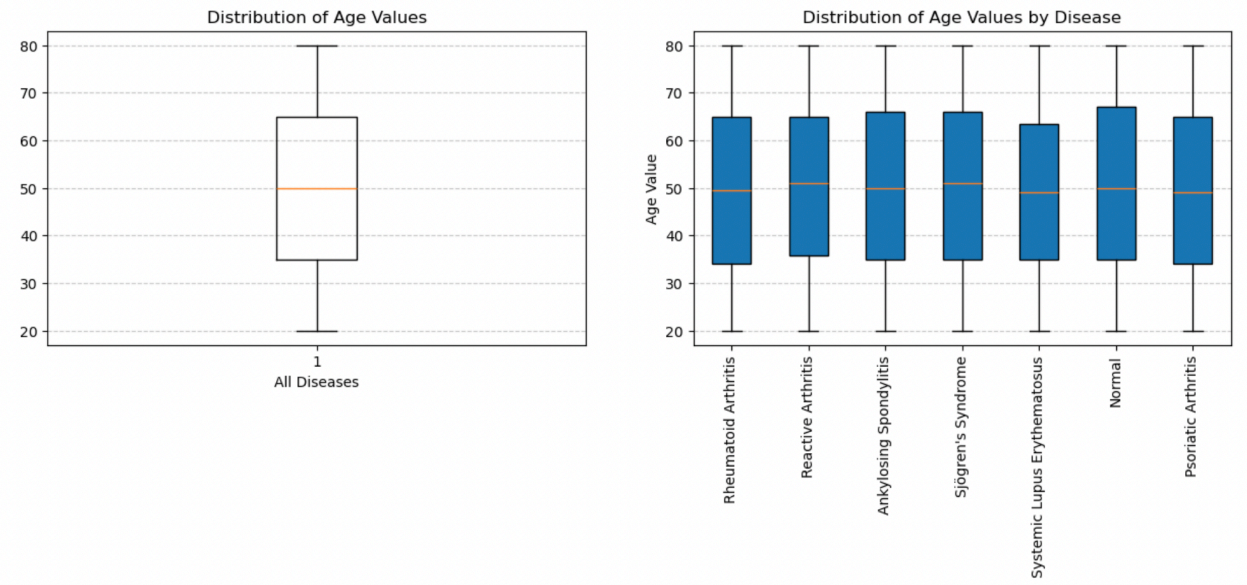
Based on the visualizations and the discussion above for each option, the Simple Imputer approach was chosen. This option retains information on which tests were not ordered for each patient, which could be useful information in itself for predicting specific diseases. See Appendix 2 for all code and additional visualizations related to addressing missing data.

Identify Trends and Relationships

The imputed dataset was analyzed to explore initial trends and relationships across predictor variables and disease classes. First, box plots were created to compare the distribution of each numerical predictor variable across the seven disease classes.

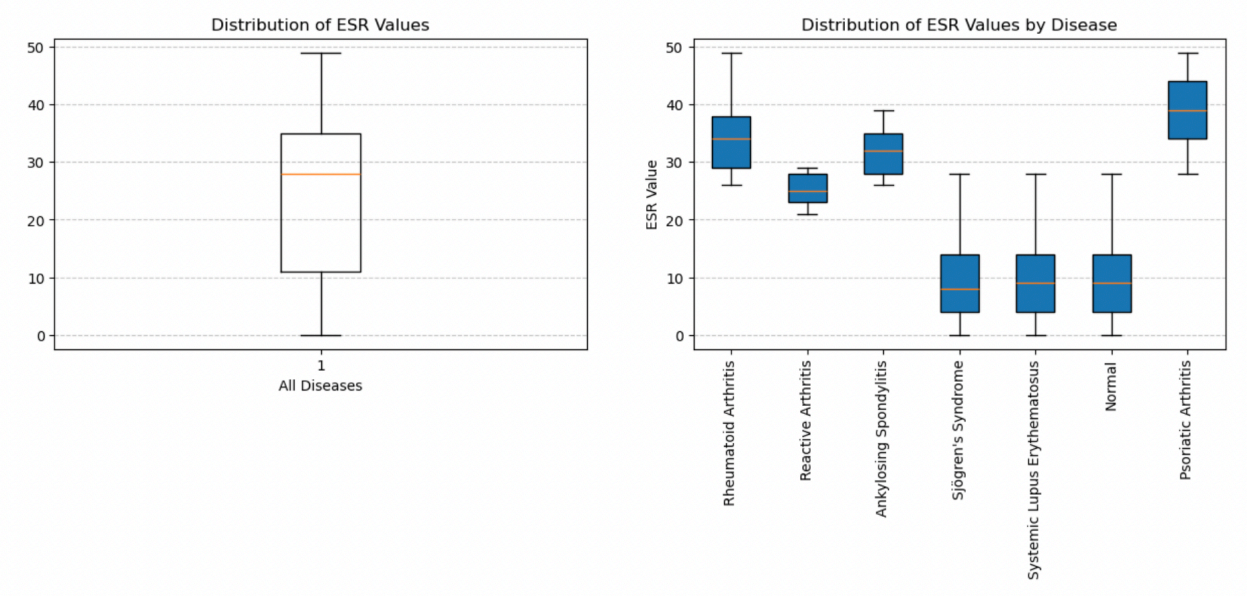
```
In [3]: # Visualize box plots of numerical features

labels = df["Disease"].unique()
num_cols = ["ESR", "CRP", "RF", "Anti-CCP",  "C3", "C4"]
cat_cols = ["HLA-B27", "ANA", "Anti-Ro", "Anti-La", "Anti-dsDNA", "Anti-Sm"]

for col in ["Age"] + num_cols:
    fig, (ax1, ax2) = plt.subplots(1,2, figsize=(15, 4), sharey=False)
    ax1.boxplot(df[col])
    ax1.set_title(f"Distribution of {col} Values")
    ax1.set_xlabel("All Diseases")
    ax1.grid(axis='y', linestyle='--', alpha=0.7)

    grouped_data = [df[col][df["Disease"] == category] for category in df["Disease"].unique()]
    ax2.boxplot(grouped_data, tick_labels=labels, patch_artist=True)
    ax2.set_title(f"Distribution of {col} Values by Disease")
    ax2.set_ylabel(f"{col} Value")
    ax2.grid(axis='y', linestyle='--', alpha=0.7)
    ax2.tick_params('x', labelrotation=90)
    plt.show()
```

Age



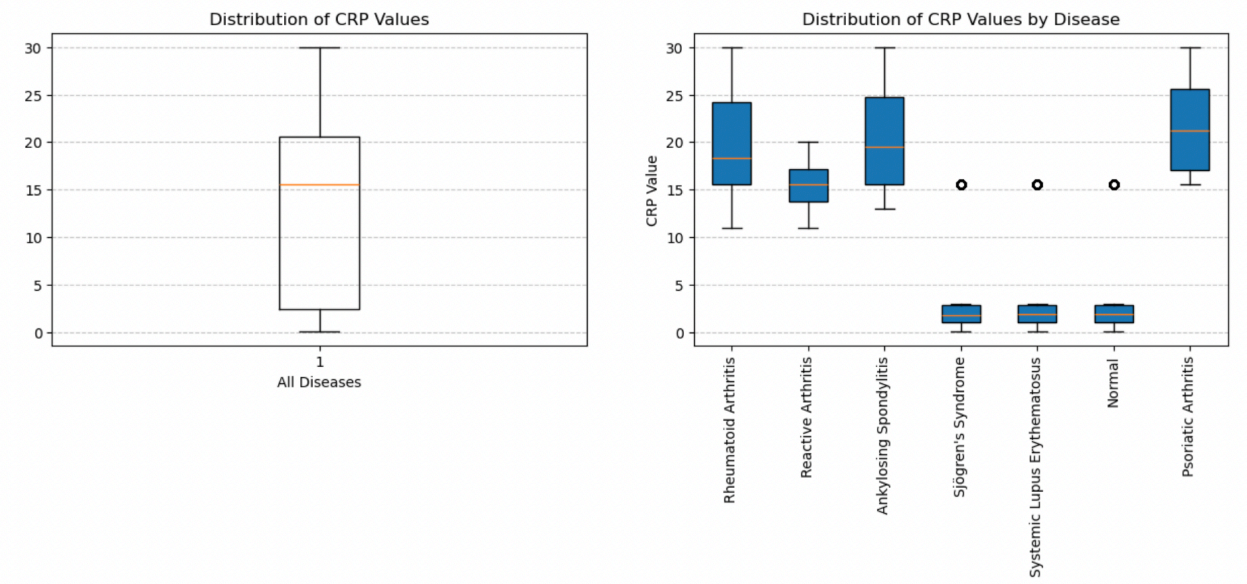
Across all disease classes, the age variable is fairly equally distributed, with a range of approximately 20 to 80 years, and the majority of patients fall between 35 and 65 years old.

ESR



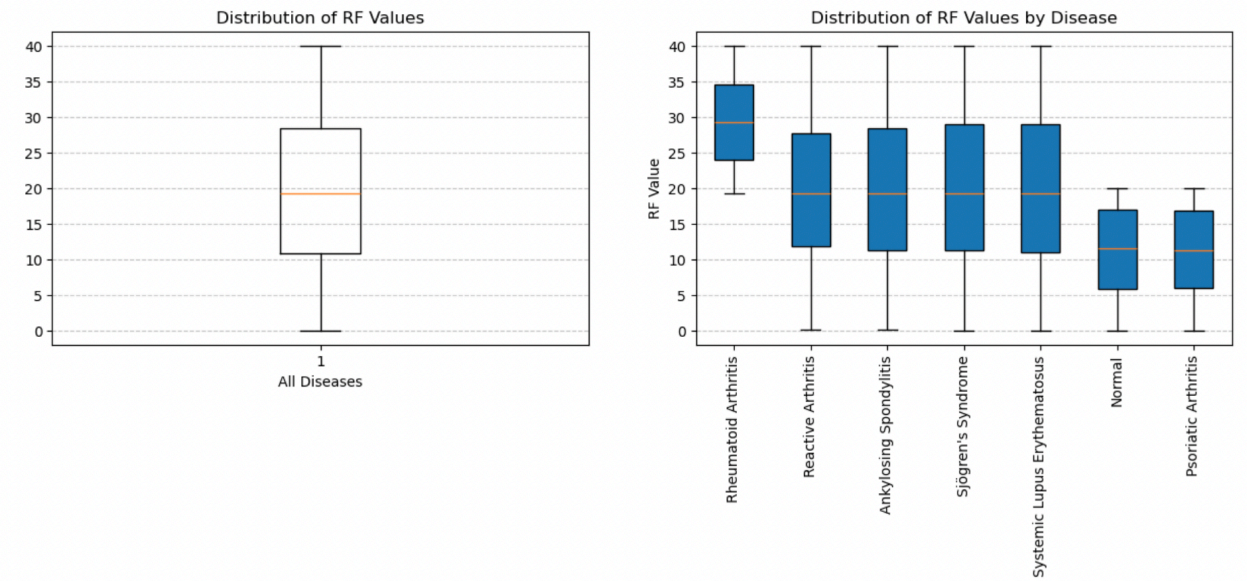
Patients with Rheumatoid Arthritis, Reactive Arthritis, Ankylosing Spondylitis, and Psoriatic Arthritis have higher ESR values than patients diagnosed with Sjogren's Syndrome, Systemic Lupus Erythematosus, and those with no disease.

CRP



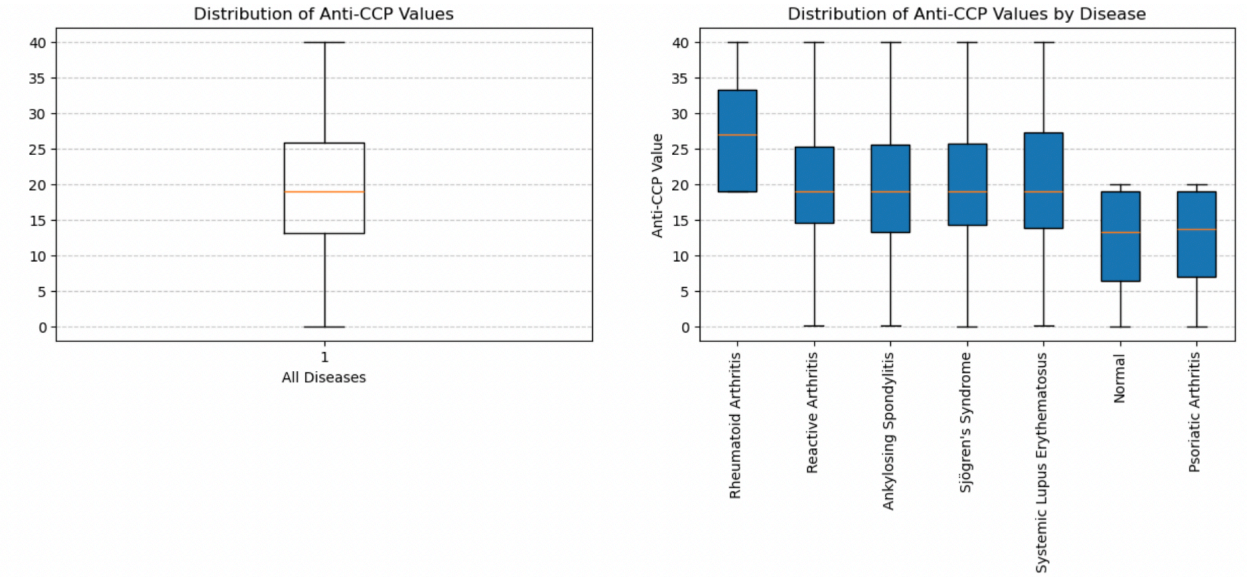
Patients have a similar distribution of CRP values as the ESR values above. The three disease classes with low CRP values have outliers near the median value for all diseases. These outliers were likely caused by imputing missing values.

RF



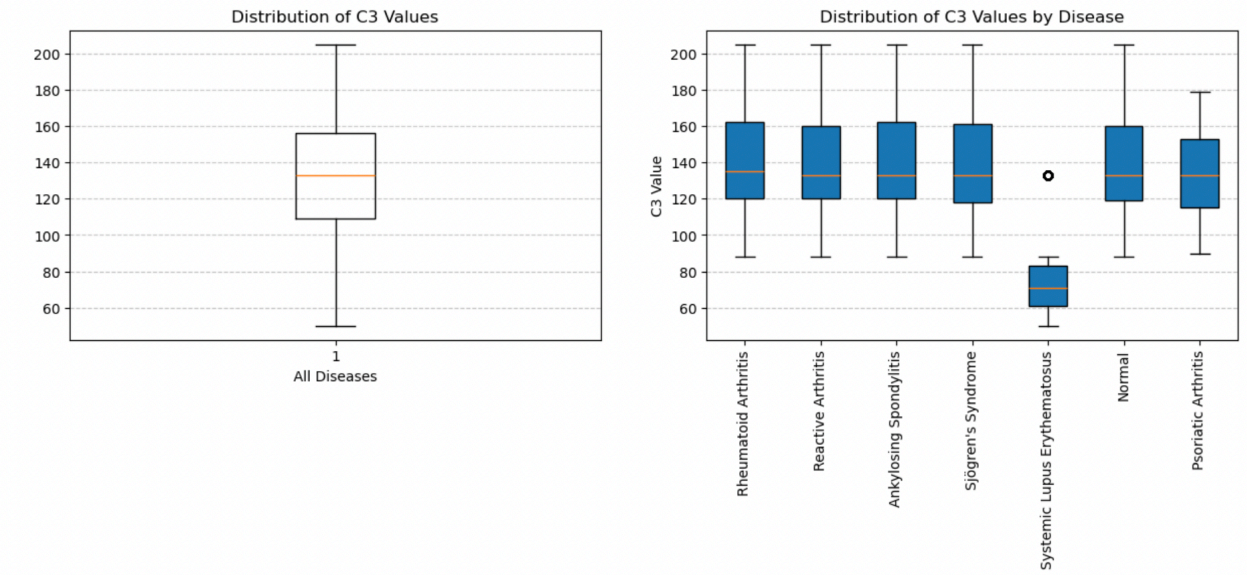
RF values are highest for those with Rheumatoid arthritis, and lowest for those with Psoriatic Arthritis or no disease.

Anti-CCP



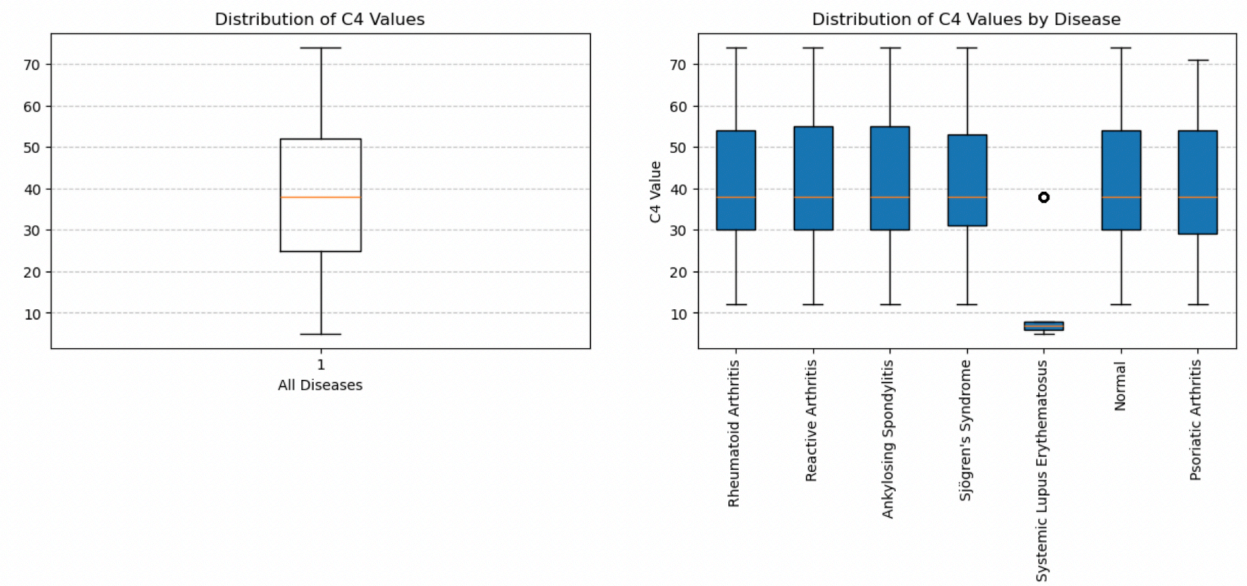
The Anti-CCP distributions across disease classes follow a similar pattern to RF distributions.

C3



Low C3 values appear to be highly specific to the Systemic Lupus Erythematosus diagnosis. The outliers are likely caused by imputing the mean for missing values.

C4

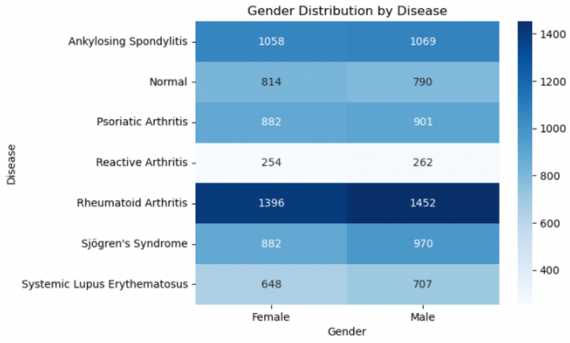


The distribution of C4 values is very similar to the C3 distribution.

Next, heatmaps were created to compare the distribution of each categorical predictor variable across the seven disease classes.

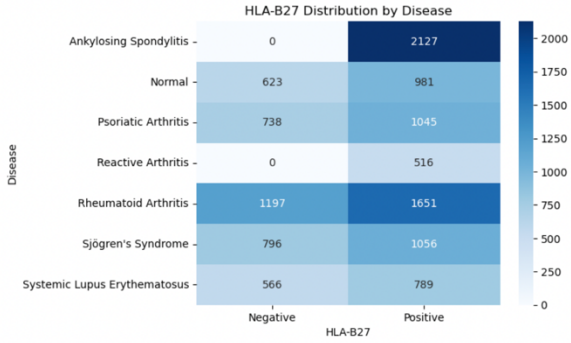
```
In [4]: # Heatmaps of categorical features and disease

for col in ["Gender"] + cat_cols:
    ct = pd.crosstab(df["Disease"], df[col])
    sns.heatmap(ct, annot=True, fmt="d", cmap="Blues")
    plt.title(f"{col} Distribution by Disease")
    plt.xlabel(f"{col}")
    plt.ylabel("Disease")
    plt.show()
```

Gender



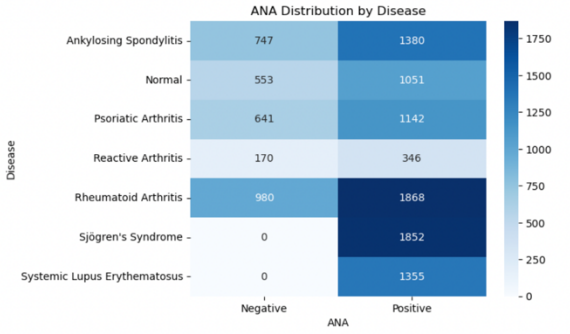
For each disease class, there are slightly more males than females represented in the data. The only class with more females than males is Normal (no disease). Females generally tend to have a higher likelihood of being diagnosed with many autoimmune and rheumatic diseases. This gender distribution may suggest underdiagnosis of women or fewer women seeking diagnosis. More research is needed to fully understand this gender distribution. (Myers, 2025).

HLA-B27



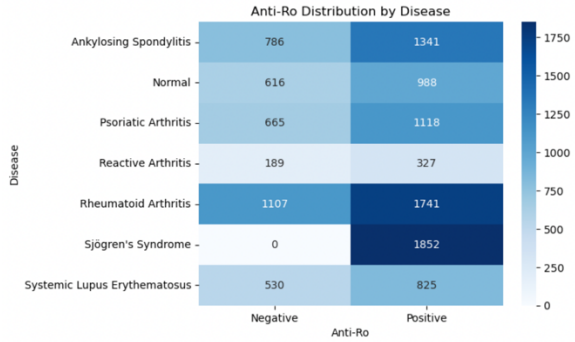
All patients diagnosed with Ankylosing Spondylitis and Reactive Arthritis are positive for the HLA-B27 marker. There are more positives than negatives across all disease classes.

ANA



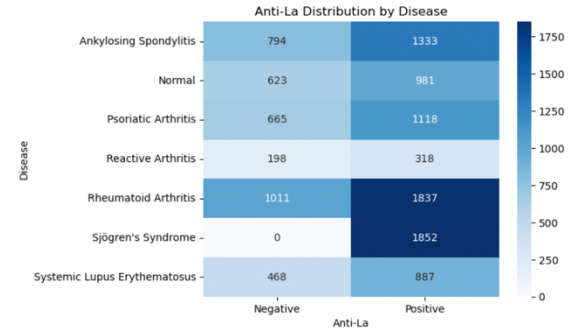
All patients diagnosed with Sjogren's Syndrome and Systemic Lupus Erythematosus are negative for ANA. Across all other disease classes, there are more positive ANA values than negative.

Anti-Ro



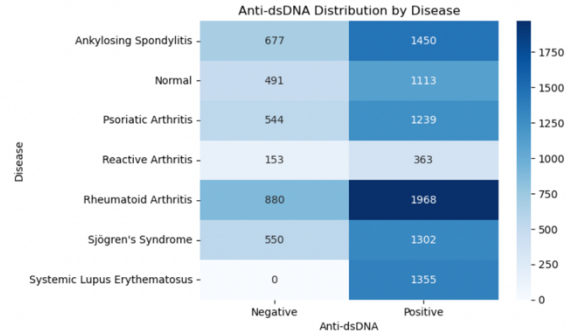
All patients diagnosed with Sjogren's syndrome are positive for Anti-Ro. Across all other classes, there are more positives than negatives.

Anti-La



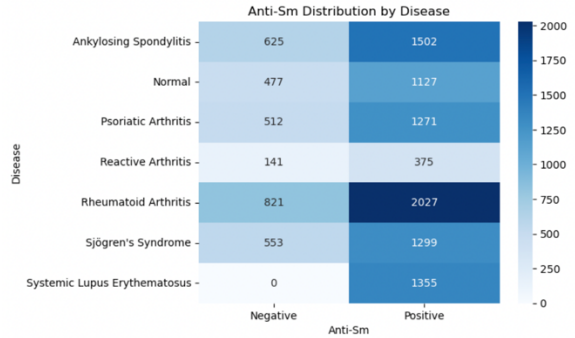
Anti-La values show a similar distribution pattern across classes as Anti-Ro.

Anti-dsDNA



All patients with Systemic Lupus Erythematosus are positive for Anti-dsDNA. Across all other classes, there are more positives than negatives.

Anti-Sm



Anti-Sm values show a similar distribution pattern across classes as Anti-dsDNA.

Summary of Patterns

The following disease associations emerge from the bivariate analysis:

Feature(s)	Strongest Disease Associations
High ESR and CRP	Rheumatoid Arthritis Reactive Arthritis Ankylosing Spondylitis Psoriatic Arthritis
High RF and Anti-CCP	Rheumatoid Arthritis
Low C3 and C4	Systemic Lupus Erythmatosus
Positive HLA-B27	Ankylosing Spondylitis Reactive Arthritis
Negative ANA	Sjogren's Syndrome Systemic Lupus Erythematosus
Positive Anti-Ro and Anti-La	Sjogren's Syndrome
Positive Anti-dsDNA and Anti-Sm	Systemic Lupus Erythmatosus

Encode Categorical Variables

The Random Forest Classification model is flexible in that it can handle a mix of numerical and categorical data types, but the categorical values need to be encoded. The target variable, Disease, was encoded using Scikit-Learn's LabelEncoder for the seven disease classes.

```
In [5]: # Use label encoding to encode target variable data
        disease_encoder = LabelEncoder()
        disease_encoder.fit(df["Disease"])

        disease_map = dict(zip(disease_encoder.classes_, range(len(disease_encoder.classes_))))
        df["Disease"] = disease_encoder.transform(df["Disease"])

        print(disease_map)
```

```
{'Ankylosing Spondylitis': 0, 'Normal': 1, 'Psoriatic Arthritis': 2, 'Reactive Arthritis': 3, 'Rheumatoid Arthritis': 4, "Sjögren's Syndrome": 5, 'Systemic Lupus Erythematosus': 6}
```

The categorical predictor variables were one-hot encoded using the Pandas get_dummies function.

```
# Use get_dummies to one-hot encode categorical predictor variable data
df = pd.get_dummies(df, columns=["Gender", "HLA-B27", "ANA", "Anti-Ro", "Anti-La", "Anti-dsDNA", "Anti-Sm"], dtype=int)
```

The resulting dataframe with imputed and encoded data includes all 12,085 samples and 34 columns. A column was added for each column with missing data with the missing indicator

during imputation, and each categorical predictor column was split into two columns, one for the positive value and one for the negative value. See Appendix 3 for the final dataset.

Out[5]:

	Age	Disease	ESR	CRP	RF	Anti-CCP	C3	C4	ESR_missing	CRP_missing	...	ANA_Negative	ANA_Positive	Anti-Ro_Negative	Anti-Ro_Positive	Anti-La_Negative	Anti-La_Positive	Anti-dsDNA_Negative	Anti-dsDNA_Positive	Anti-Sm_Negative	Anti-Sm_Positive
0	70	4	39.0	18.6	34.2	29.9	133.0	27.0	0.0	0.0	...	1	0	0	1	1	0	0	1	0	1
1	39	4	26.0	21.7	35.5	28.9	100.0	66.0	0.0	0.0	...	0	1	0	1	0	1	0	1	0	1
2	36	4	41.0	15.6	21.3	21.3	158.0	12.0	0.0	0.0	...	1	0	0	1	0	1	1	0	0	1
3	35	4	43.0	23.4	26.0	39.0	119.0	41.0	0.0	0.0	...	0	1	0	1	0	1	0	1	0	1
4	37	4	30.0	15.6	38.1	30.8	144.0	49.0	0.0	1.0	...	1	0	0	1	1	0	0	1	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
12080	32	2	36.0	17.0	14.5	16.1	133.0	32.0	0.0	0.0	...	0	1	1	0	0	1	0	1	0	1
12081	36	2	43.0	15.6	17.7	13.5	133.0	41.0	0.0	1.0	...	1	0	1	0	0	1	0	1	1	0
12082	20	2	31.0	28.8	4.8	5.8	133.0	38.0	0.0	0.0	...	0	1	0	1	1	0	1	0	1	0
12083	33	2	36.0	15.6	19.2	9.5	96.0	52.0	0.0	1.0	...	0	1	0	1	0	1	0	1	0	1
12084	48	2	32.0	26.9	7.2	13.6	178.0	38.0	0.0	0.0	...	0	1	0	1	1	0	0	1	0	1

12085 rows × 34 columns

Split Data into Training, Validation, and Testing Sets

The data was split into three sets using Scikit-Learn's train_test_split function. The training set includes 60% of the samples, while the validation and testing sets include 20% of the samples each. The value counts of each disease in each dataset were analyzed to ensure there was sufficient representation of each disease class in each set. There are at least 100 instances of the least-represented class (Reactive Arthritis) in each set. An advantage of using this tool to split the data is that it retains a similar proportion of each class within each set. However, a disadvantage is that it is unable to balance the classes across each set, due to the imbalanced nature of the original dataset. This would need to be addressed by balancing the original dataset.

In [20]:
```
# Split data into training, validation, and testing datasets

X = df.drop(["Disease"], axis=1)
y = df["Disease"]

# 1st split - 20% test
X_temp, X_test, y_temp, y_test = train_test_split(X, y, test_size = 0.2)
# 2nd split - 60% train, 20% validate
X_train, X_val, y_train, y_val = train_test_split(X_temp, y_temp, test_size = 0.25)

# Save training, validation, and testing datasets to csv
train_df = pd.concat([X_train, y_train], axis = 1)
val_df = pd.concat([X_val, y_val], axis = 1)
test_df = pd.concat([X_test, y_test], axis = 1)

train_df.to_csv('training_data.csv', index=False)
val_df.to_csv('validation_data.csv', index=False)
test_df.to_csv('testing_data.csv', index=False)
print("Train, validation, and test data sets saved to CSV")

# Examine distribution of outcome variable in training, validation, and testing data sets
print(f"Train: {train_df["Disease"].value_counts()}")
print(f"Validate: {val_df["Disease"].value_counts()}")
print(f"Test: {test_df["Disease"].value_counts()}")
```

```
Train: Disease
4    1747
0    1274
5    1075
2    1060
1     963
6     829
3     303
Name: count, dtype: int64
Validate: Disease
4    522
0    440
2    380
5    373
1    340
6    250
3    112
Name: count, dtype: int64
Test: Disease
4    579
0    413
5    404
2    343
1    301
6    276
3    101
Name: count, dtype: int64
```

Analysis

Random Forest Classifier

The predictive model was developed using the Random Forest Classifier from Scikit-Learn.

Random Forest Justification

The Random Forest Classifier is a supervised machine learning algorithm. It is an ensemble tree-based learning approach, meaning that each predicted class is derived from a collection of predictions from multiple individual trees. In this approach, the predictor features are run through multiple decision trees to determine a class label for each tree (in this case, a disease label). After the classes are predicted from each tree, majority voting is used to assign the final prediction label to the sample.

One advantage of using the Random Forest Classification technique for this analysis is that it can handle categorical as well as continuous feature data. In a comprehensive review of multiple machine learning techniques for medical diagnostic purposes, Random Forest was found to have high accuracy with high computing efficiency compared to other ensemble methods.

> "The RF classifier is one of the most successfully implemented ensemble learning techniques which have proved very popular and powerful for high-dimensional classification and skewed problems in pattern recognition and ML. It offers the benefit of computing efficiency and improves the accuracy of predictions without considerably increasing calculation costs. Based on these characteristics, most of the researchers recorded the highest value of accuracy for the prediction of diseases." (Ray & Chaudhuri, 2021).

However, a disadvantage of this technique is that it requires imputation of all missing values. As discussed above, missing data is a significant issue in this medical dataset. There are other ensemble models that could be used without as much pre-processing and without first imputing values for missing data points. Gradient Boosting and XG Boost are two techniques that natively handle missing values without the need for imputation (Ashtari, 2024).

Random Forest Implementation

First, an initial Random Forest Classifier model was built, fit to the training data using the default hyperparameters, and validated using the validation data. This initial model had an overall accuracy of 83.33% across all disease classes. Next, the model was optimized through hyperparameter tuning. Scikit-Learn's RandomizedSearchCV tool was used to test various combinations of hyperparameters to find the most accurate combination for the final model. See Appendix 4 for the initial model code and evaluation, and hyperparameter tuning results.

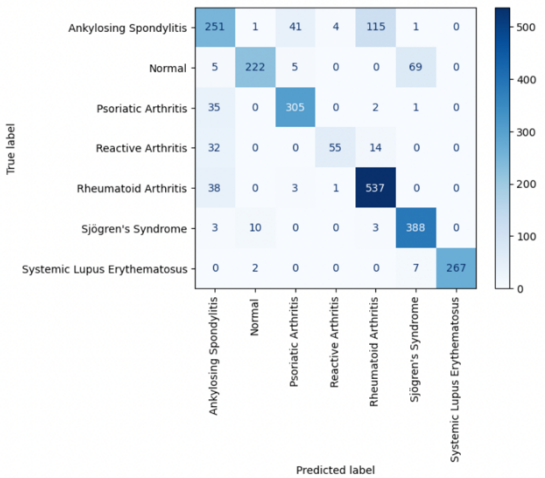
After the best hyperparameter values were determined, the tuned model was fit to the training data and evaluated using the test data. As a result of hyperparameter tuning, the accuracy of the tuned model improved slightly to 83.78%.

```
In [28]: # Refit model with best hyperparameters

rfc_tuned_model = RandomForestClassifier(
    random_state=42,
    class_weight=best.class_weight,
    n_jobs=1,
    n_estimators=best.n_estimators,
    max_features=best.max_features,
    max_depth=best.max_depth,
    min_samples_leaf=best.min_samples_leaf,
    criterion=best.criterion
)
rfc_tuned_model.fit(X_train, y_train)
y_pred = rfc_tuned_model.predict(X_test)

y_pred_proba = rfc_tuned_model.predict_proba(X_test)

# Evaluate tuned model
# accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Tuned Model Accuracy: {accuracy * 100:.2f}%")

# ROC-AUC score
roc_auc = roc_auc_score(y_test, y_pred_proba, multi_class="ovr")
print(f"Tuned Model ROC-AUC score: {roc_auc * 100:.2f}%")

# classification report (precision, recall, F1)
report = classification_report(y_test, y_pred, target_names=disease_encoder.classes_)
print(f"Tuned Model Classification Report")
print(report)

# confusion matrix
matrix = confusion_matrix(y_test, y_pred, labels=rfc_tuned_model.classes_)
matrix_vis = ConfusionMatrixDisplay(confusion_matrix=matrix,
                                    display_labels=disease_encoder.inverse_transform(rfc_model.classes_))
matrix_vis.plot(cmap=plt.cm.Blues)
plt.xticks(rotation=90)
plt.show()
```

Random Forest Model and Evaluation Output

The tuned model's performance is reflected in the following output metrics. These will be interpreted and discussed in the final section.

```
Tuned Model Accuracy: 83.78%
Tuned Model ROC-AUC score: 98.42%
Tuned Model Classification Report
                          precision    recall  f1-score   support

      Ankylosing Spondylitis       0.69      0.61      0.65       413
                      Normal       0.94      0.74      0.83       301
        Psoriatic Arthritis        0.86      0.89      0.88       343
        Reactive Arthritis         0.92      0.54      0.68       101
      Rheumatoid Arthritis         0.80      0.93      0.86       579
        Sjögren's Syndrome         0.83      0.96      0.89       404
Systemic Lupus Erythematosus       1.00      0.97      0.98       276

                  accuracy                           0.84      2417
                 macro avg       0.86      0.80      0.82      2417
              weighted avg       0.84      0.84      0.83      2417
```



Permutation Test

The model's predictions were evaluated using Scikit-Learn's permutation_test_score function.

Permutation Test Justification

This test is able to compare the accuracy of the model's predictions on the test data against a number of permutations of samples with randomly shuffled disease labels. The results of this test evaluate how well the model performs compared to a model making predictions with labels assigned by random chance. An advantage of this test is that it is able to evaluate the model's predictive ability holistically, rather than looking only at a certain feature or set of features' predictive influence on a target variable.

However, a disadvantage is that the test is very costly to implement. To run the test, one must specify a number of permutations to run. The larger the number, the stronger the evaluation, but every increase in the number of permutations requires a lot more computing power and time to run the model that many times. In this analysis, only 100 permutations were used, but this required significantly more computing power than the training, hyperparameter tuning, or testing of the model itself.

Permutation Test Implementation

The permutation test returns an overall observed accuracy score (calculated across all permutations), the individual permutation scores, and a p-value.

```
In [29]: # Hypothesis testing – Permutation test

score, permutation_scores, pvalue = permutation_test_score(
    estimator=rfc_tuned_model,
    X=X_test,
    y=y_test,
    scoring="accuracy",
    n_permutations=100,
    n_jobs=1,
    random_state=42
)

print(f"Observed accuracy: {score}")
print(f"Permutation p-value: {pvalue}")
```

Permutation Test Output

The permutation test generated the following output for observed accuracy and p-value, which will be interpreted and discussed in the next section.

```
Observed accuracy: 0.828726280307308
Permutation p-value: 0.009900990099009901
```

Data Summary and Implications

The goal of this study was to create a predictive model to support early diagnosis of difficult-to-diagnose rheumatic and autoimmune diseases. In order to accomplish this, the predictive power of a specific set of variables needed to be demonstrated. This generated the following research question: To what extent do blood test result variables (including ESR, CRP, RF, Anti-CCP, HLA-B27, ANA, Anti-Ro, Anti-La, Anti-dsDNA, Anti-Sm, C3, and C4) predict a specific autoimmune disease diagnosis? The null hypothesis was that the blood test result variables alone can not accurately predict a disease, while the alternate hypothesis is that the blood test result variables statistically significantly predict a disease.

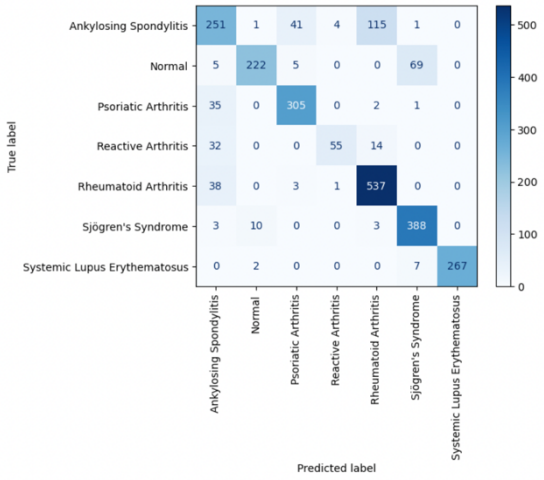
Results Interpretation

To evaluate the tuned model's performance around the project goal, we can interpret the accuracy, ROC_AUC score, classification report scores, and confusion matrix. The accuracy of the model's predictions across all classes was 83.78%, with an ROC-AUC score of 98.42%. A high ROC-AUC score paired with a lower accuracy score suggests that the model has strength in ranking the probability of each disease class for a particular test sample, but at the same time may land on the wrong final label when it takes the majority vote from the ensemble of decision trees.

```
Tuned Model Accuracy: 83.78%
Tuned Model ROC-AUC score: 98.42%
Tuned Model Classification Report
                              precision    recall  f1-score   support

      Ankylosing Spondylitis       0.69      0.61      0.65       413
                      Normal       0.94      0.74      0.83       301
        Psoriatic Arthritis        0.86      0.89      0.88       343
        Reactive Arthritis         0.92      0.54      0.68       101
     Rheumatoid Arthritis          0.80      0.93      0.86       579
        Sjögren's Syndrome         0.83      0.96      0.89       404
Systemic Lupus Erythematosus       1.00      0.97      0.98       276

                  accuracy                           0.84      2417
                 macro avg       0.86      0.80      0.82      2417
              weighted avg       0.84      0.84      0.83      2417
```

The F1-scores on the classification report help to illustrate why the accuracy percentage is lower. The score for each individual disease class ranges from 65% for Ankylosing Spondylitis to 98% for Systemic Lupus Erythematosus. Therefore, the model makes highly accurate predictions on some diseases, and has mediocre performance on others.

The confusion matrix helps illustrate the specific strengths and weaknesses of the model by disease class. The strongest performer, Systemic Lupus Erythematosus (SLE), had an F1 score of 98%. The confusion matrix shows that the model correctly predicted 267 patients had SLE, while it predicted a total of 9 false negatives (misdiagnosing 7 patients with Sjogren's Syndrome



and 2 patients as normal). The worst performer was Ankylosing Spondylitis, with an F1-score of 65%. The model predicted a number of false negatives for Ankylosing Spondylitis patients, with predicted diagnoses of Rheumatoid Arthritis and Psoriatic Arthritis. In addition, the model predicted a lot of false positive results, misdiagnosing patients who actually have Psoriatic, Rheumatoid, and Reactive Arthritis with Ankylosing Spondylitis instead.

The results of the permutation test have a similar accuracy to the model's performance on the test data, coming in at approximately 83% across all classes. The p-value result of 0.0099 supports the alternate hypothesis that the variables the model is built around are predictive in classifying disease labels. This means that there is less than 1% chance that the model's predictive power could be explained by chance. This is very strong evidence that the set of predictor variables, which include only objective blood test results, can statistically significantly predict the target labels for rheumatic and autoimmune diseases.

```
Observed accuracy: 0.828726280307308
Permutation p-value: 0.009900990099009901
```

Implications and Limitations of the Analysis

In summary, this analysis supports the alternate hypothesis that the blood test result variables statistically significantly predict a disease. This is an important finding given the context of the study. These are diseases that are difficult to diagnose clinically, and there is evidence that bias can get in the way of accurate diagnoses. However, despite the high p-value supporting this alternate hypothesis, the primary goal of the study, the creation of a predictive model, was less successful. The overall accuracy of around 84%, with individual disease

accuracies as low as 65%, suggests that more work needs to be done before a disease prediction model can be used successfully in a clinical or patient education context.

The main limitations that are likely impacting the performance of the predictive model are the issue of missing values and the imbalanced representation of the disease classes in the original dataset. These issues are not unique to this study and are, in fact, very common in machine learning applications in the medical field.

> "Generally, healthcare datasets are highly imbalanced in nature. Classification of these datasets results in erroneous prediction and inaccurate accuracies. Another very common characteristic of healthcare datasets is a large number of missing data values for multiple features." (Ray & Chaudhuri, 2021).

Thus, more work needs to be done around addressing these two issues before implementing a diagnostic model in clinical practice. The recommended course of action is to engage in further experimentation with the model towards a goal of 85% accuracy across each individual disease class before deploying the model to support clinical decision-making.

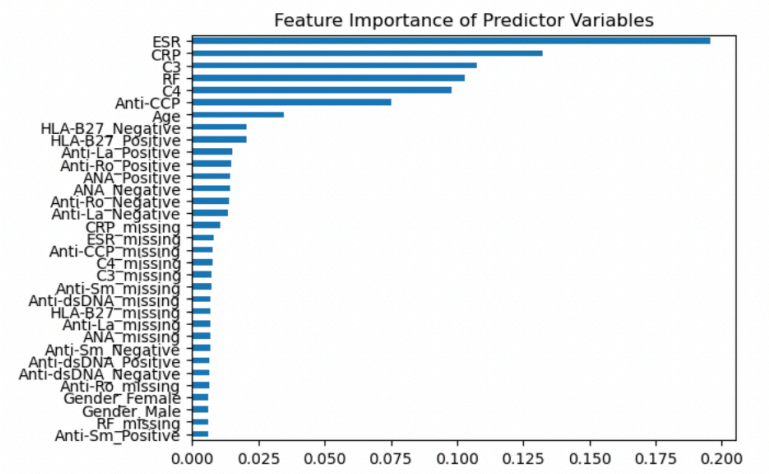
Recommendations for Further Analysis

Overall, this initial study shows promise towards achieving the goal of supporting the diagnosis of autoimmune and rheumatic diseases using a predictive model. Three directions for further analysis are detailed below, ordered by the amount of capacity it might take to pursue each.

1. Refine and/or redefine the predictor variables

There are several pieces of evidence explored in this analysis to suggest that the variables in the original dataset are predictive of specific diseases (the p-value, the findings from bivariate analysis, and the high predictive accuracy of certain diseases). Assuming these are the right blood test variables, more work can be done through feature engineering to determine the best way to use the variables in the predictive model. For example, the current model consumes a mix of numeric and categorical blood test values. However, the numeric values could be converted to categorical (positive/negative) values using the normal reference ranges for each test, and additional columns could be added to indicate whether the result is positive or negative.

This visualization shows the feature importance, or relative influence, of each variable in the predictive model. The top seven features are all of the numeric features, followed by all of the categorical features (positive result columns, negative result columns, and missing indicator columns). More exploration is needed to determine whether converting the numerical features to categorical features would level the playing field, and perhaps help the model to use more of the information from the categorical features. One approach would be to make every column binary (positive/negative), while another possible approach could be to also include a missing option for each variable



(positive/negative/missing). Both approaches should be experimented with to determine the best structure of the existing data.

2. Use a more complex imputation technique, or a tool that natively addresses missing values

After determining the best structure of the existing data, more exploration could be done around the benefits and limitations of different imputation techniques and modeling tools. Austin et. al. recommend the Multiple Imputation technique for imputing values in medical data sets.

> "MI imputes multiple values for each missing value. This results in the creation of multiple complete data sets in which the missing values have been filled in with plausible values. The analysis of scientific interest is then conducted separately in each of these complete data sets, and the results are pooled across the imputed data sets. In this way, MI allows the user to explicitly incorporate the uncertainty about the true value of imputed variables." (Austin et. al., 2021).

There is evidence that Random Forest is a strong performer in the context of medical classification and diagnosis, and the associated challenges in the medical field (Ray & Chaudhuri, 2021). However, implementing a Gradient Boosting model or XG Boost model without using imputation for missing values could yield different results in accuracy (Ashtari, 2024). Either the Multiple Imputation approach or using a different model with native support for missing values has the potential to lead to a more accurate predictive model.

3. Generate a more robust dataset for analysis

Finally, if the steps above yield promising results, it could be worth investing resources into collecting a more robust dataset for the diagnosis of autoimmune and rheumatic diseases. The current dataset is very limited in scope (all patients are from a single country and hospital). In addition, the bivariate analysis around gender and disease distribution was surprising in that

more males were diagnosed with diseases that are more common in females, and more females were diagnosed as normal. This dataset may, in fact, reflect and reinforce some of the clinical bias around gender, in particular, that the analysis is trying to avoid (Myers, 2025). Therefore, a more valid and accurate model might be derived from a dataset that represents a broader population from multiple settings (including international representation with patients from multiple clinical settings), includes diversity of demographic features (such as age, gender and race), and has a more balanced distribution of diseases (intentionally increasing the number of samples from the more rare disease classes).

In summary, while this Random Forest Classification model's accuracy is not high enough to warrant deployment, the analysis shows promise that the diagnosis of rheumatic and autoimmune diseases could be supported by machine learning models. The further development of this model, focused only on objective blood test results, has the potential to mitigate some of the bias that currently exists in the diagnosis of these diseases.

References

Ashtari, H. (2024). *XGBoost vs. Random Forest vs. Gradient Boosting: Differences | Spiceworks*. Spiceworks. https://www.spiceworks.com/tech/artificial-intelligence/articles/xgboost-vs-random-forest-vs-gradient-boosting/

Austin, P. C., White, I. R., Lee, D. S., & van Buuren, S. (2020). Missing Data in Clinical research: a Tutorial on Multiple Imputation. *Canadian Journal of Cardiology*, *37*(9). https://doi.org/10.1016/j.cjca.2020.11.010

Creative Commons. (2019). *Creative Commons — CC0 1.0 Universal*. Creativecommons.org. https://creativecommons.org/publicdomain/zero/1.0/

Dhafar Hamed Abd; Mohammed Fadhil Mahdi; Arezoo Jahani. (2025). "Rheumatic and autoimmune diseases dataset", https://doi.org/10.7910/DVN/VM4OR3, Harvard Dataverse, V3

Mahdi, M. F., Jahani, A., & Abd, D. H. (2025). Diagnosis of rheumatic and autoimmune diseases dataset. *Data in Brief*, *60*, 111623. https://doi.org/10.1016/j.dib.2025.111623

Miller, F. W. (2023). The increasing prevalence of autoimmunity and autoimmune diseases: an urgent call to action for improved understanding, diagnosis, treatment, and prevention. *Current Opinion in Immunology*, *80*(102266), 102266. https://doi.org/10.1016/j.coi.2022.102266

Myers, Emma. (2025). *Silent Struggles: How Autoimmune Diseases in Women Are Overlooked – North Carolina Schweitzer Fellowship*. Ncschweitzerfellowship.org. https://ncschweitzerfellowship.org/silent-struggles-how-autoimmune-diseases-in-women-are-overlooked/

Ray, A., & Chaudhuri, A. K. (2021). Smart healthcare disease diagnosis and patient management: Innovation, improvement and skill development. *Machine Learning with Applications*, *3*, 100011. https://doi.org/10.1016/j.mlwa.2020.100011

Scikit-Learn. (2025). *sklearn.ensemble.RandomForestClassifier — scikit-learn 0.20.3 Documentation*. Scikit-Learn.org. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

List of Appendices

The following appendices are included with this report:

Appendix 1: Excel file with original dataset

Appendix 2: Jupyter notebook with missing data experimentation

Appendix 3: Cleaned and imputed dataset

Appendix 4: Jupyter notebook with model code and evaluation

Appendix 5-7: Training, validation, and testing datasets