Reuben W. Martor

DSC 680- Applied Data Science
Project 1 Milestone 2

Submitted 9/15/2024

Title: Developing a Symptom-Based System for Disease Diagnosis and Precaution Recommendations in Resource-Limited Settings

Title: Developing a Symptom-Based System for Disease Diagnosis and Precaution Recommendations in Resource-Limited Settings

1. Business Problem

Accurate disease diagnosis is essential for timely treatment. However, in many developing countries and rural areas, limited access to diagnostic resources can delay or prevent accurate diagnoses, leading to incorrect treatments and adverse outcomes. In such environments, healthcare providers often rely solely on symptoms and patient history, which can increase the risk of misdiagnosis.

This project aims to address this issue by developing a machine learning-based system that leverages symptom data to assist healthcare providers in making more accurate diagnoses. The system will also recommend appropriate precautions and treatment options, providing a solution that is both accessible and effective in resource-constrained settings.

2. Background/History

Delays in diagnosing certain health conditions can lead to poorer outcomes, especially in time-sensitive medical scenarios. This issue is exacerbated in developing regions with limited access to advanced diagnostic tools. In such settings, medical professionals often rely on subjective patient reports and physical symptoms to guide treatment decisions, which may lead to misdiagnosis.

By applying machine learning to analyze symptom patterns and historical patient data, this project seeks to improve diagnostic accuracy. The goal is to enable faster, more informed

decision-making that leads to better patient outcomes, particularly in resource-constrained environments.

3. Data Explanation

Datasets were sourced from Kaggle, and significant effort will be made to clean, preprocess, and split them into training, validation, and test sets for model development and evaluation. The goal is to ensure the data is reliable, diverse, and globally applicable.

The datasets used include:

DATASETS

- **data:** A disease and its associated symptoms

  Contains the diseases and up to 17 associated symptoms.

  Columns: Disease, Symptom_1, Symptom_2, …, Symptom_17

- **disease_desc:** A disease or health condition and its definition

  Contains the description/definition of each disease.

  Columns: Disease, Description.

- **symptom_severity :** symptoms associated with a health condition and its severity or ranking or weight

  Contains the severity weight of each symptom.

  Columns: Symptom, weight.

- **symptom_precaution :** a health condition and recommended precautions

  Contains the precautions for each disease.

  Columns: Disease, Precaution_1, Precaution_2, Precaution_3, Precaution_4

4. Methods

Data cleaning and preparations were performed in preparation of model development. After data preparation, various models were evaluated, including Logistic Regression, Support Vector Classification (SVC), K-Nearest Neighbors (KNN), DecisionTreeClassifier, and Random Forest. With cross validation and hyper tunning, all models performed excellently well thereby qualifying for fitting the data. Optionally if time allows, a user-friendly interface will be created for healthcare providers to easily interact with the model, making the system practical in real-world settings.

5. Analysis

I used Principal Component Analysis (PCA) to determine the features with the most predicting power. The cumulative variance explained by the principal components shows how much of the total dataset variance is captured as I include more components. From the PCA, the first 40 components or features were degerming factors for prediction, predicting approximately 97% of the variation disease classification. At 77th feature, 99% of the variance is explained. For completeness sick and medical certainty, I included all features due to their presumptive significance. The analysis focused on the accuracy of each model in predicting disease diagnoses based on symptoms. Performance metrics such as precision, recall, and accuracy were used to assess the models.

Post-analysis:

I elected to the RandomForestClassifier because of its:

1. Robustness: It averages across many decision trees, which makes it less prone to overfitting compared to a single decision tree.
2. Scalability: It can handle larger datasets better than models like KNeighborsClassifier and DecisionTreeClassifier.
3. Generalization: It typically generalizes better to new data due to its ensemble nature. the selected model will be integrated into a system to provide symptom-based disease diagnosis recommendations.

Analysis of Random Classifier- Classification report

1. Precision: Indicates how many of the predicted positive cases were correct. All conditions have a precision score of 1.00, meaning that all predictions made by the model were accurate for each class.
2. Recall: Represents how many of the actual positive cases were correctly predicted. Here, all conditions also have a recall score of 1.00, signifying that the model was able to identify all relevant cases for each condition.
3. F1-score: This is the harmonic mean of precision and recall, providing a single metric for model performance. Since both precision and recall are 1.00 for all classes, the F1-score is also 1.00 across the board.
4. Support: Refers to the number of actual instances for each class. This varies per condition, indicating that the number of samples used to evaluate each class differs.

5. Accuracy: Overall accuracy is 1.00, meaning that the model predicted every case correctly across all conditions.
6. Macro average: This is the average of the precision, recall, and F1 scores, calculated for each class without considering class imbalance. It is 1.00, showing that the model performs perfectly across all conditions.
7. Weighted average: This is the same as the macro average, but it accounts for class support (the number of samples in each class), also resulting in 1.00.

This Random Forest classifier achieved perfect performance in this specific dataset with respect to precision, recall, and F1-score.

6. Conclusion

By using machine learning models, this project aims to provide healthcare providers in resource-limited settings with timely and reliable information for disease diagnosis and treatment recommendations. This approach has the potential to improve patient outcomes, especially in environments where diagnostic resources are scarce.
All models tested performed excellent. However, the Random Forest Classifier selected for further processing.

7. Assumptions

- The datasets used are representative of symptom patterns in real-world clinical settings., however, the symptoms are limited and do not encompass all practical symptoms and descriptions in actual clinical settings
- The models will be used against symptoms already captured
- For generalizations, more symptoms are needed to train the model

8. Limitations

- The quality and diversity of the datasets may limit the system's global applicability.

Symptoms may mimic differently in different genders. For example, chest pain which may demonstrate cardiac arrest may present differently in male vs female. Additionally, Age may affect medical presentation, but the training data did not account for age variations.

- The system's recommendations are based on symptom data alone, without access to more definitive diagnostic tests such as blood work or imaging. In a real medical setting, patient history may be of key component in presumptive diagnosis.

9. Challenges

- Time constraint-I find it challenging to modify the model due limited time meeting project deadline.
- Data cleaning and transformation: Cleaning and transforming the datasets from categorical to numerical was difficulty but key to ensure reliable model performance.
- Designing a user-friendly interface: Making the project more practical was one of my goals. With limited programming experience, I found it taxing to design an interface for implementation for healthcare workers to easily navigate in resource-constrained environments.
- Ensuring that the model provides consistent and reliable recommendations across different clinical contexts- This remains the challenge for the model as I look forward to testing from external symptoms and diseases beyond the dataset available for this study.

10. Future Uses/Additional Applications

The system could be expanded to incorporate additional data sources, such as patient's vital signs or local epidemiological data, to improve diagnosis accuracy. In the future, this technology could be applied to other healthcare challenges in resource-constrained settings, such as predicting disease outbreaks or optimizing healthcare supply chains.

11. Recommendations

- I need to make more effort to increase the robustness of the datasets by incorporating data from various sources and regions to ensure global applicability.
- Future updates and validations of these machine learning models should be a priority to ensure they remain accurate over time.

12. Implementation Plan

1. **Phase 1:** Data cleaning and preprocessing.
2. **Phase 2:** Model training, validation, and hyperparameter tuning.
3. **Phase 3:** Development of a user interface for healthcare workers.
5. **Phase 5:** Iterative refinement and full deployment: in the future

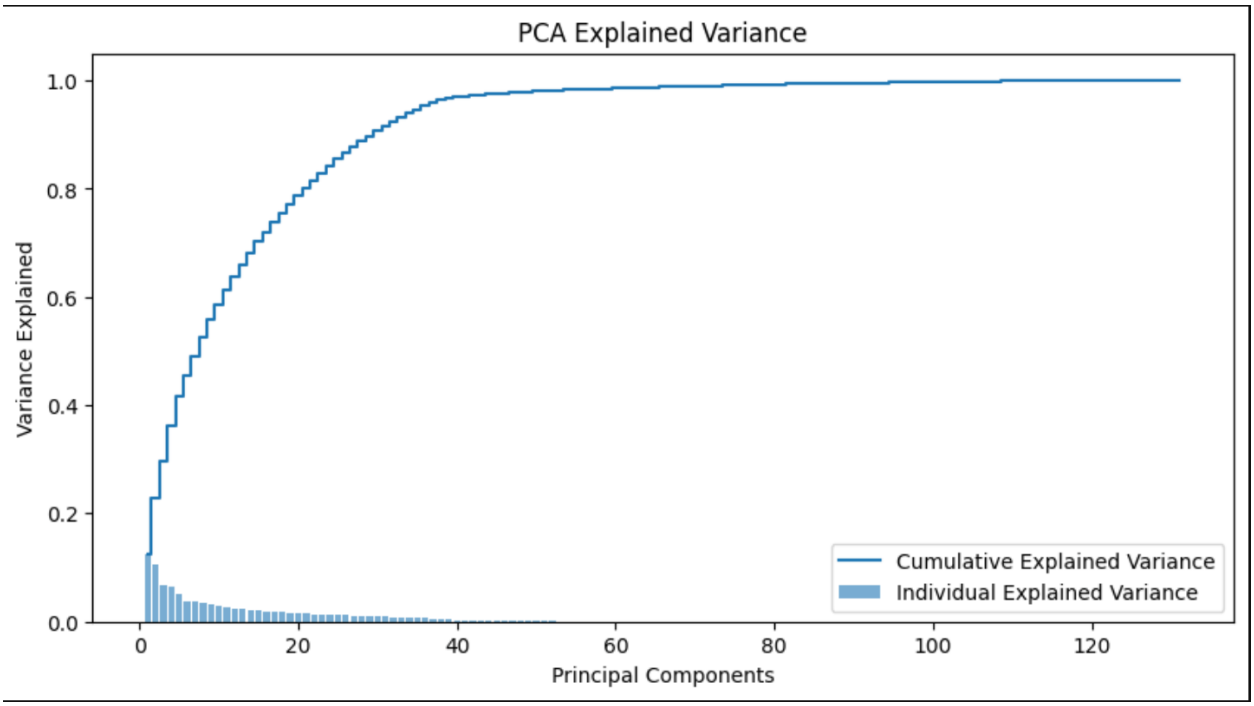
13. Ethical Assessment

Several ethical considerations are paramount for this project:
Data security and privacy remains a strong pillar for HIPAA. Obtaining patient data for future testing seem challenging. Datasets may be difficult to obtain without special commitment or process. Informed consent, transparency in data access and usage needs to be prioritized must be emphasized when using this system. Disclaimer: This system is for academic use only currently. It should not be used to replace professional medical advice.

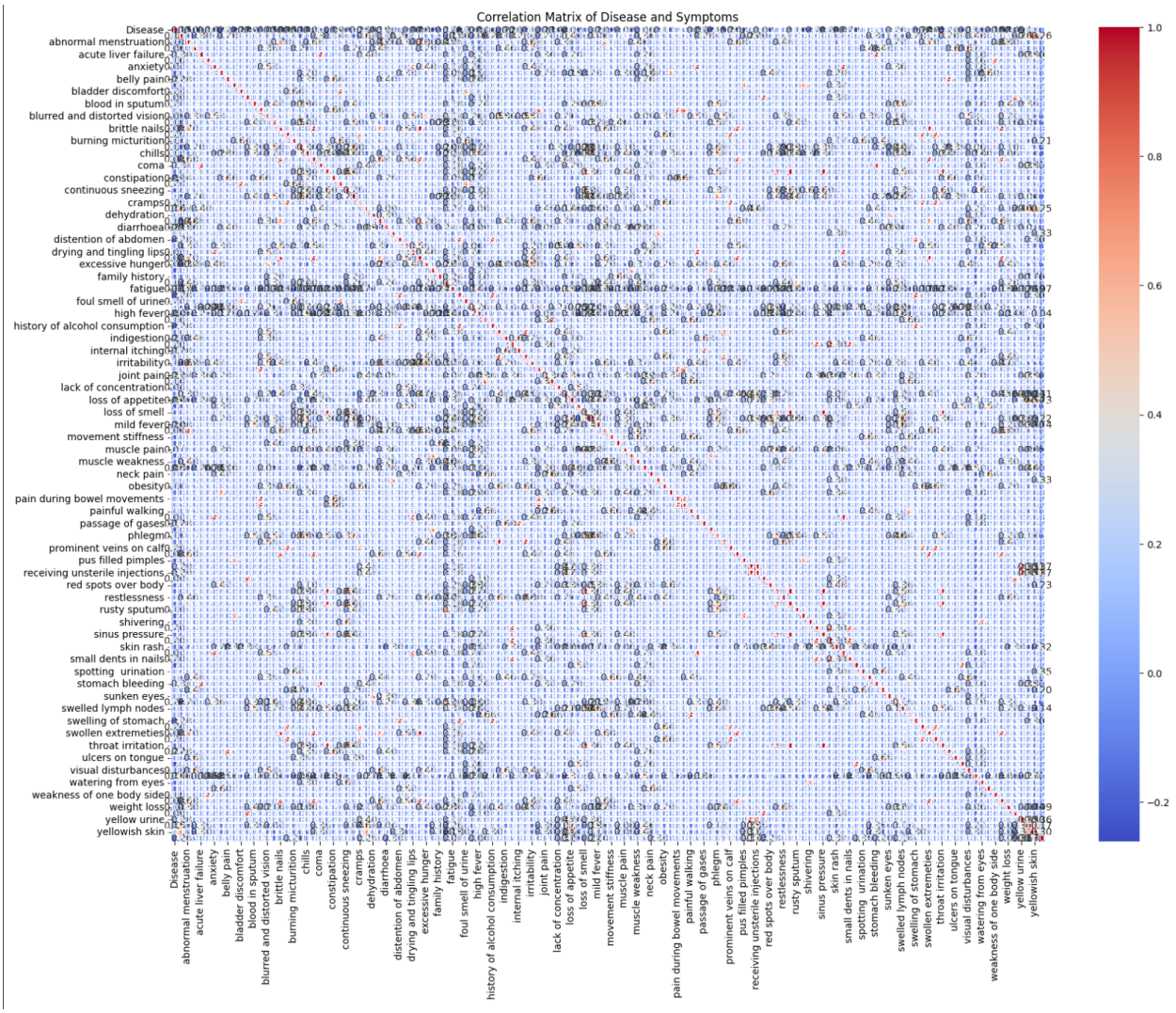
Ten (10) Potential questions audience may ask:

1. What is the primary objective of this project?
2. What kind of data did you use for this model, and where was it sourced from?
3. How did you preprocess the data before using it for model training?
4. Which machine learning models did you evaluate, and why did you choose RandomForestClassifier?
5. How does the Random Forest Classifier help avoid overfitting?
6. What performance metrics were used to evaluate the model?
7. Did you encounter any challenges during the model development phase?
8. How did you handle symptoms that were not present in every disease case in the dataset? 9. What role does Principal Component Analysis (PCA) play in this project?
10. How do you ensure the system's global applicability, especially in different geographical regions?

Appendix A: PCA



Appendix B Correlation Matrix



Appendix C GridSearchCV/Cross Validation performance score

GridSearchCV/Cross validation/Hyperparameter tunning performance results

```
# We can now apply hyper tunning
selectModel(Models, hyper_params)
```

```
LogisticRegression(max_iter=10000)
{'C': [1, 5, 10, 20]}
------------------------------------------------------------------
SVC()
{'kernel': ['linear', 'poly', 'rbf', 'sigmoid'], 'C': [1, 5, 10, 20]}
------------------------------------------------------------------
KNeighborsClassifier()
{'n_neighbors': [2, 3, 5, 10]}
------------------------------------------------------------------
RandomForestClassifier(random_state=42)
{'n_estimators': [10, 20, 50, 100]}
------------------------------------------------------------------
DecisionTreeClassifier(random_state=42)
{'criterion': ['gini', 'entropy'], 'max_depth': [None, 10, 20, 30], 'min_samples_split': [2, 10, 20], 'min_samples_leaf': [1, 5, 10]}
------------------------------------------------------------------
```

	Model used	Highest Score	Best hyperparameters
0	LogisticRegression	1.0	{'C': 1}
1	SVC	1.0	{'C': 1, 'kernel': 'linear'}
2	KNeighborsClassifier	1.0	{'n_neighbors': 2}
3	RandomForestClassifier	1.0	{'n_estimators': 10}
4	DecisionTreeClassifier	1.0	{'criterion': 'gini', 'max_depth': None, 'min_...

Appendix D: Code snapets:

clean datasets using functions

```
# Define a function to clean datasets by stripping whitespace, replacing unwanted characters, and filling NaN values with 'None'

def clean_datasets(datasets_info):
    cleaned_datasets = {}

    # Define a helper function to clean each DataFrame
    def clean_dataframe(df):
        # Strip whitespace, replace unwanted characters, and fill NaN values with 'None'
        df = df.applymap(lambda x: x.strip().replace('_', ' ') if isinstance(x, str) else x)
        df = df.fillna('None')
        return df

    # Loop through each dataset in the provided datasets_info dictionary
    for name, dataset in datasets_info.items():
        cleaned_datasets[name] = clean_dataframe(dataset)

    return cleaned_datasets


cleaned_datasets = clean_datasets({
    "Main Dataset": dataset,
    "Disease Description": disease_description,
    "Symptom Precaution": symptom_precaution,
    "Symptom Severity": symptom_severity
})


# Display the cleaned "Main Dataset" as an example
cleaned_datasets['Main Dataset'].head()
```

	Disease	Symptom_1	Symptom_2	Symptom_3	Symptom_4	Symptom_5	Symptom_6	Symptom_7	Symptom_8	Symptom_9	Symptom_10	Symptom
0	Fungal infection	itching	skin rash	nodal skin eruptions	dischromic patches	None	None	None	None	None	None	N
1	Fungal infection	skin rash	nodal skin eruptions	dischromic patches	None	None	None	None	None	None	None	N
2	Fungal infection	itching	nodal skin eruptions	dischromic patches	None	None	None	None	None	None	None	N
3	Fungal infection	itching	skin rash	dischromic patches	None	None	None	None	None	None	None	N
4	Fungal infection	itching	skin rash	nodal skin eruptions	None	None	None	None	None	None	None	N

Gather all symptom columns then extract unique symptom names from all symptom columns

```
symptom_columns = [col for col in cleaned_datasets['Main Dataset'].columns if 'Symptom' in col]

# Extract unique symptom names from all symptom columns in the dataset
unique_symptoms = set()
for col in symptom_columns:
    unique_symptoms.update(dataset[col].dropna().unique())


# convert the set to list and sort ( excluding 'nan' if any)
unique_symptoms = sorted(list(unique_symptoms))
# initialize a binary dataframe to hold binary features
symptom_features = pd.DataFrame(0, index=dataset.index, columns=unique_symptoms)

# populate the binary features dataFrame: 1 if symptom is present, 0 otherwise
for index, row in dataset.iterrows():
    for col in symptom_columns:
        symptom = row[col]
        if pd.notna(symptom) and symptom in symptom_features.columns:
            symptom_features.at[index, symptom]= 1
transformed_data = pd.concat([dataset['Disease'], symptom_features], axis=1)
transformed_data.columns = transformed_data.columns.str.replace('_', ' ')
```

Now combine symptom features with the disease label

```
transformed_data.head()
```

	Disease	abdominal pain	abnormal menstruation	acidity	acute liver failure	altered sensorium	anxiety	back pain	belly pain	blackheads	...	watering from eyes	weakness in limbs	weakness of one body side	weight gain	weight loss
0	Fungal infection	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
1	Fungal infection	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
2	Fungal infection	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
3	Fungal infection	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
4	Fungal infection	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0

5 rows × 132 columns

Perform PCA analysis

```
]: # extract features
   feature_df= transformed_data.drop(columns=['Disease']) # features
   target= transformed_data['Disease'] # Target variables

   pca = PCA()
   pca.fit(feature_df)

   # Get variance ratios
   variance_ratios = pca.explained_variance_ratio_
   cumulative_variance = np.cumsum(variance_ratios)

   counter = 0

   # Print feature names, explained variance, and cumulative variance for each feature
   # Print header with aligned columns
   print(f"{'Symptoms':<30}{'Individual Variance':<25}{'Component Count':<15}{'Cumulative Variance':<25}\n",'_'*90)

   # Print feature names, explained variance, and cumulative variance for each feature
   for feature_name, var_ratio, cum_var in zip(feature_df.columns, variance_ratios, cumulative_variance):
       counter += 1
       print(f"{feature_name:<30}{var_ratio:<28.5f}{counter:<20}{cum_var:<22.5f}")
```

```
Symptoms                      Individual Variance      Component CountCumulative Variance
__________________________________________________________________________________________
 abdominal pain               0.12421                  1                   0.12421
 abnormal menstruation        0.10520                  2                   0.22941
 acidity                      0.06844                  3                   0.29785
 acute liver failure          0.06545                  4                   0.36330
 altered sensorium            0.05211                  5                   0.41541
 anxiety                      0.03872                  6                   0.45413
 back pain                    0.03655                  7                   0.49068
 belly pain                   0.03432                  8                   0.52500
 blackheads                   0.03312                  9                   0.55812
 bladder discomfort           0.02857                  10                  0.58669
 blister                      0.02579                  11                  0.61249
 blood in sputum              0.02413                  12                  0.63662
 bloody stool                 0.02365                  13                  0.66027
 blurred and distorted vision 0.02146                  14                  0.68173
 breathlessness               0.02027                  15                  0.70200
 brittle nails                0.01840                  16                  0.72039
 bruising                     0.01766                  17                  0.73806
```


References:

Khorasani, M., & Abualrob, A. (2022). *Web application development with Streamlit: Develop and deploy secure and scalable web applications*. Apress. [Kindle version].

Wells, F. (2019). *Web app development with Streamlit: A comprehensive guide to creating interactive data dashboards, advanced visualization and integrating machine learning* (1st ed.). [Kindle version]

Fuentes, A. (n.d.). *Hands-on predictive analytics with Python: Master the complete predictive analytics pipeline using Python*. [Kindle version].

Itachi9604. (n.d.). *Disease symptom description dataset* [Data set]. Kaggle. Retrieved from https://www.kaggle.com/datasets/itachi9604/disease-symptom-description-dataset