Heart Disease Classification: K-Nearest-Neighbors

Zachary Weinfeld, Dillon Murphy, and Nick Patrick

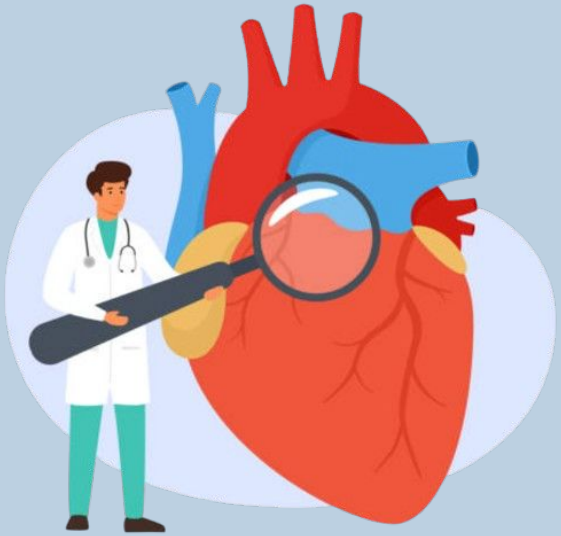
Overview

1. Motivation
2. Data
3. K-Nearest-Neighbors
4. Hyperparameter Tuning
5. Results and Conclusion

1. Motivation

Motivation

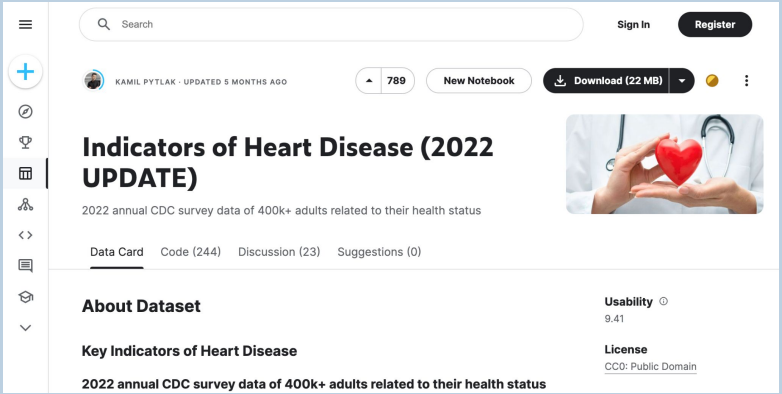
- More than 800,000 Americans die each year from Cardiovascular Disease
- This makes up around ⅓ of all deaths in the US
- Effectively predicting if a patient has heart disease can potentially save lives



2. Data

Data

- This Kaggle dataset contains over 300,000 rows and 18 features
- The target column is `HeartDisease' which indicates whether or not the subject is diagnosed with heart disease



https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease/data

Data Cleaning (Before)

HeartDisease	BMI	Smoking	AlcoholDrinking	Stroke	PhysicalHealth	MentalHealth	DiffWalking	Sex
No	16.60	Yes	No	No	3.0	30.0	No	Female
No	20.34	No	No	Yes	0.0	0.0	No	Female
No	26.58	Yes	No	No	20.0	30.0	No	Male
No	24.21	No	No	No	0.0	0.0	No	Female
No	23.71	No	No	No	28.0	0.0	Yes	Female
Yes	28.87	Yes	No	No	6.0	0.0	Yes	Female
No	21.63	No	No	No	15.0	0.0	No	Female
No	31.64	Yes	No	No	5.0	0.0	Yes	Female
No	26.45	No	No	No	0.0	0.0	No	Female
No	40.69	No	No	No	0.0	0.0	Yes	Male

319795 rows × 18 columns

Issues:

- Missing data
- Yes/no binary data
- Binned data (age)
- Categorical Data (health condition)
- Non-normalized data

Issues and Solutions

- **Missing data**: removed rows with missing data (less than 10,000 rows removed)
- **Yes/no binary data**: converted “yes” to 1 and “no” to 0
- **Binned data**: mapped each bin to its lower bound (eg: age “55 to 60” -> 55)
- **Categorical Data**:
    - Mapped *ordinal* data to integers (eg: health condition=”Good” -> 4)
    - Mapped *nominal* data to dummy variables (eg: race -> dummy columns)
- **Non-normalized data**: divided every column by its maximum value, so all data is mapped to the interval [0, 1]
    - This helps increase the accuracy of KNN

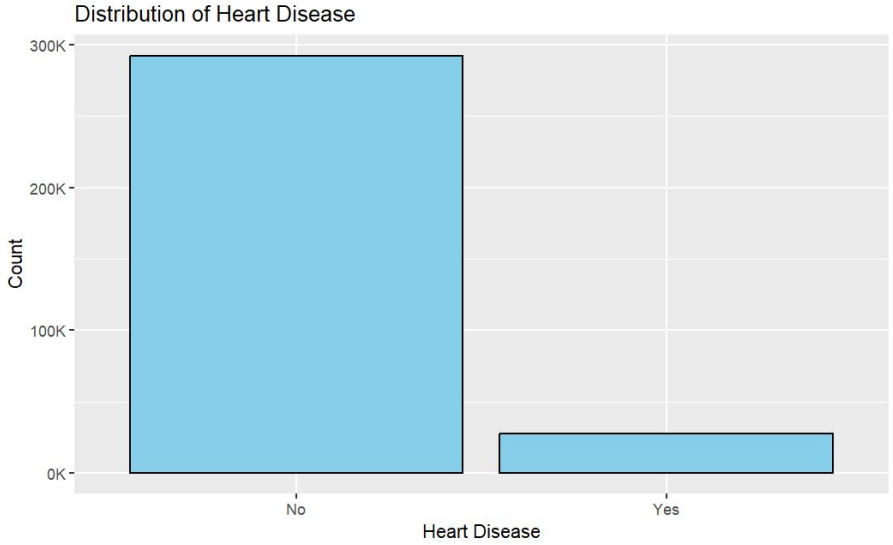
Data Cleaning (After)

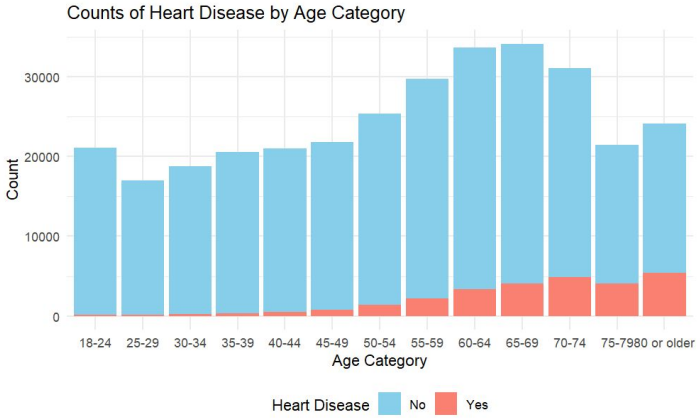
HeartDisease	BMI	Smoking	AlcoholDrinking	Stroke	PhysicalHealth	MentalHealth	DiffWalking	Sex
0.0	0.175013	1.0	0.0	0.0	0.100000	1.0	0.0	0.0
0.0	0.214444	0.0	0.0	1.0	0.000000	0.0	0.0	0.0
0.0	0.280232	1.0	0.0	0.0	0.666667	1.0	0.0	1.0
0.0	0.255245	0.0	0.0	0.0	0.000000	0.0	0.0	0.0
0.0	0.249974	0.0	0.0	0.0	0.933333	0.0	1.0	0.0
1.0	0.304375	1.0	0.0	0.0	0.200000	0.0	1.0	0.0
0.0	0.228044	0.0	0.0	0.0	0.500000	0.0	0.0	0.0
0.0	0.333579	1.0	0.0	0.0	0.166667	0.0	1.0	0.0
0.0	0.428993	0.0	0.0	0.0	0.000000	0.0	1.0	1.0
1.0	0.361624	1.0	0.0	0.0	1.000000	0.0	1.0	1.0

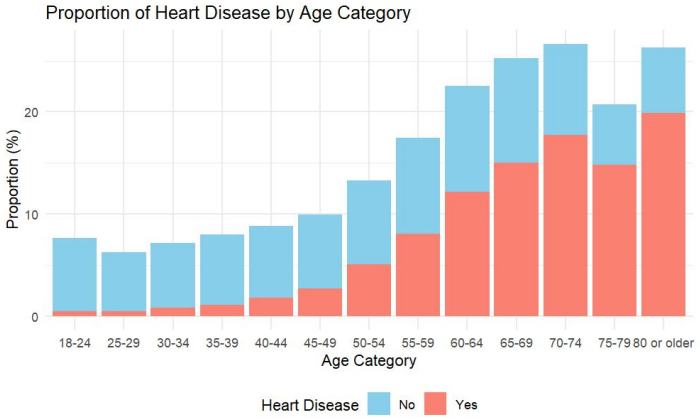
310455 rows × 23 columns

Data Visualizations

One potential issue with the data is how unbalanced it is. There is a much larger proportion of people without heart disease than people with heart disease in the data



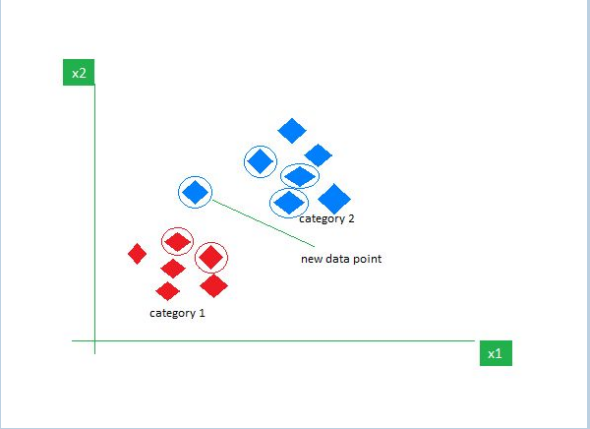




3. K-Nearest-Neighbors

K-Nearest-Neighbors Algorithm

- For each datapoint in the test set, find the K Nearest Neighbors, where “near” is defined by a distance function (in our case, Euclidean distance)
- Test different values of K to determine the optimal hyperparameter



$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 * (\frac{Precision * Recall}{Precision + Recall})$$

Performance Metrics

- Every class has its own F1 score
- Macro F1 Score: unweighted average

4. Hyperparameter Tuning

Hyperparameters

Value of K

- The K Nearest Neighbors algorithm needs to specify how many neighbors (K) to look at

Number of Features, N

- Some features should not be included in the model, as they adversely affect accuracy
- In general, these are the features with low or negative correlation with the target variable

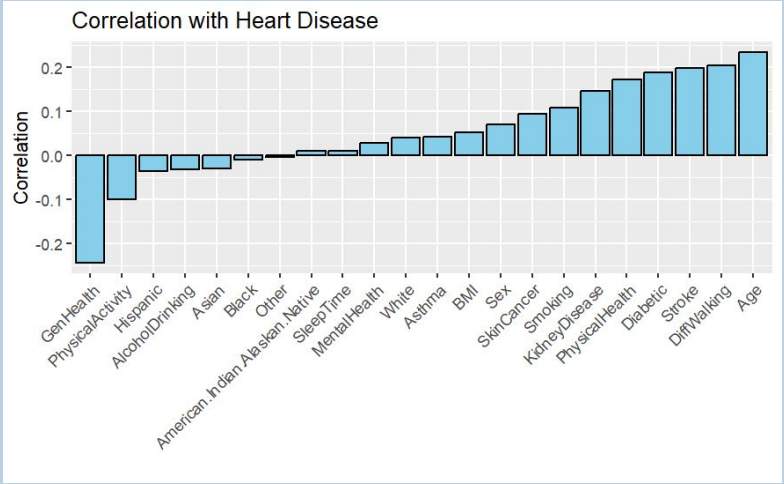
Hyperparameter 1: Value of K

- Tested every other value of `K` from 2 to 30 on a *subset* (10,000 rows) of the dataset
- There is a clear maximum at **`k=14`** neighbors

```
(k, macroF1)
(14,0.5623249299719889)
(12,0.5578947368421052)
(6,0.5529266111837128)
(10,0.5497611776681545)
(20,0.5483073411943704)
(16,0.5442530307173457)
(18,0.5442530307173457)
(4,0.5365332688510056)
(8,0.5286195286195287)
(24,0.5277860622328263)
(26,0.5277860622328263)
(22,0.5261354523583874)
(2,0.5251576834862386)
(28,0.5040565629848165)
(30,0.5040565629848165)
```

Hyperparameter 2: Correlation-based Number of Features

- Columns with higher correlation with the target variables are better predictors
- Determine the correlation of each variable with the target variable
- Include the top `N` in the model



Hyperparameter 2: Number of Features, N

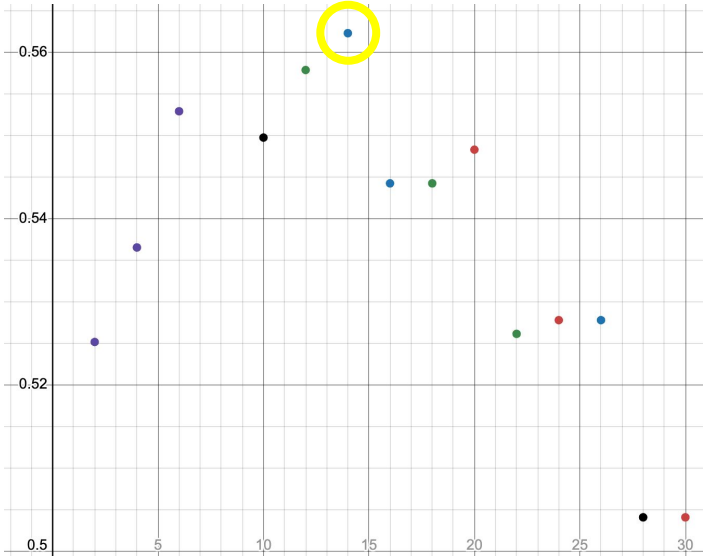
- Only want to use features that are highly positively correlated with target
- Tested value of `N` from 1 to 20 on *subset* (10,000 rows) of the dataset
- There is a clear maximum at **`N=13`** features

(N, macroF1)

```
(13,0.7120526532291238)
(14,0.6927803379416283)
(15,0.6927803379416283)
(16,0.6927803379416283)
(17,0.6927803379416283)
(18,0.6927803379416283)
(4,0.6842105263157895)
(12,0.6756756756756757)
(19,0.6756756756756757)
(20,0.6603260869565217)
(9,0.6464159434265511)
(3,0.6453900709219859)
(7,0.6159754224270353)
(10,0.6159754224270353)
(11,0.6159754224270353)
(8,0.6036036036036035)
(2,0.5670995670995671)
(6,0.5670995670995671)
(5,0.5475113122171946)
(1,0.4791666666666667)
```

Hyperparameters, Summarized

Value of K



Number of Features, N

5. Results and Conclusion

Results

80:20 train/test split on *entire* dataset (310,455 rows) using optimal hyperparameters

Confusion Matrix

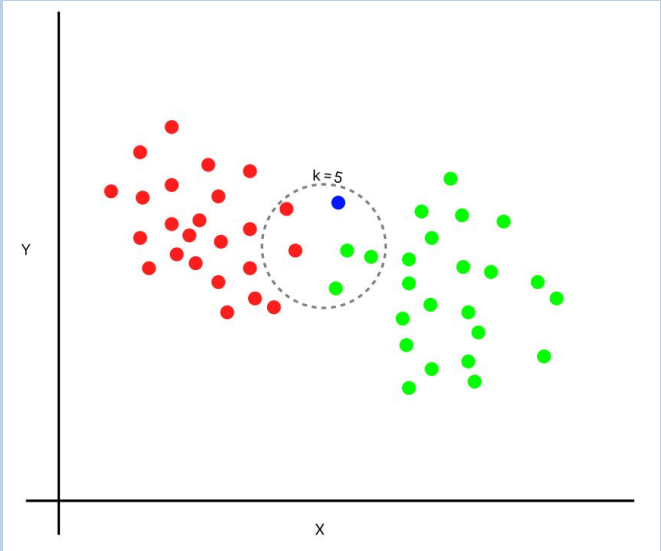
Actual \ Predicted	1	0
1	573	724
0	4,633	56,161

Metrics

Macro F1	Accuracy
0.5654	0.9137

Summary

- The optimal value of `K` was 14 neighbors
- The optimal of `N` was 13 features
- This gave our model a macro F1 score of 0.71 on a small subset of our data, and 0.57 on a larger subset of the data



Reflection

Obstacles

- First attempt: decision tree was very hard to implement
- Logistic regression performed worse than KNN
- Imbalanced target variable
- Our dataset likely did not provide the best predictor variables

Takeaways

- Models are much faster on the server
- Using scala and implementing KNN proved to not be too difficult, but making our model perform well was
- Even after tuning our hyperparameters, KNN did not perform very well