Report

This project consists of Supervised and Unsupervised learning on different data sets to show the different use cases for each machine learning method.

Supervised Learning Portion

In this portion of the project I used Supervised learning. The data I used came from Kraggle, and it was predicting who has heart disease and who doesn't based on different factors. I used different classifiers to see which one produced the best accuracy rate. The classifiers I used were

- Decision Tree
- KNN
- Naive Bayesian

Overview of Algorithms Used and Explained + Pseudo Code/Real Code

KNN (K Nearest Neighbors) is a specific classification that is used to predict labeled data (like all supervised methods). However, KNN is used mostly when predicting either a "Yes" or "No" situation. KNN also has a method that finds the best parameters to get the best accuracy rate called GridSeach and is used to find the best parameters. Decision Tree on the other hand is a different classification that does not do well with continuous data and however, provides a visual representation of nodes that branch out to different other nodes depending on questions.(Hillier, 6). Naive Bayesian is another classification that is used for more discrete labels compared to continuous/numerical labels.(Vadapalli,11) I used all 3 classifications to see the difference in accuracy between them and see which one is used. (Aasim, 1).

Pseudo code: for KNN

```
Split data into testing_set and training_set
Set model with parameters
Fit the model with training_set
Predict the testing_set
Predict the training_set
```

Actual Code:

```
Xtrain, Xtest, ytrain, ytest = train_test_split(heart_df, heart_disease,
test_size=0.2, random_state=13)
```

```
model = KNeighborsClassifier(n_neighbors = 5,
metric='euclidean',weights='uniform') # This the first attempt with random
weight, metric and neighbors
model.fit(Xtrain,ytrain)
ypred = model.predict(Xtest)
y_pred_training = model.predict(Xtrain)
print("Testing accuracy:",accuracy_score(ytest,ypred))
print("Training accuracy:",accuracy_score(ytrain,y_pred_training))
print(classification_report(ytest,ypred))
```

You split the data into a training set and testing set. You then create the model with different parameters. You then fit the model with the training set, once you do that you use the model to predict the testing set and training set. Then you can see the accuracy score from both the testing set and training set with the classification report that gives you the recall and precision.

Analysis/Summary

- **Decision Tree**:
    - **Training accuracy:** .99 = 99%
    - **Testing accuracy**: .86 = 86%
    - We can see that overfitting is occurring here since the training is **WAY** higher then the testing accuracy
- **Naive Bayesian:**
    - **Training accuracy:** .74 = 74%
    - **Testing accuracy**: .74 = 74%
- **K Nearest Neighbors:**
    - **Before Grid Search:**
        - **Training Accuracy:** .915 = 92 %
        - **Testing accuracy** .913 = 91%
- **AFTER Grid Search:**
    - **Training accuracy**: .915 = 92% (Rounded up)
    - **Testing accuracy**: .913 = 91%
    - This is strange that we got the same accuracy, and I that is something I wanted to investigate but regardless this is still a good accuracy.

Based on all the accuracies we can see that K Nearest Neighbors is the best one. It’s not overfitting like **Decision Tree** and it's not low compared to **Naive Bayesian.** Which makes sense, because as stated above some of these classifications are not meant for this type of data I used for the project. However, accuracy isn’t everything in machine learning. We should really care about recall and precision.

Side Discussion about Precision and Recall

Scenirco: You are trying to train a model that predicts oranges and apples. You are mostly predicting apples.

Precision - "Number of apples predicted correctly by the model / Number of apples and oranges predicted correctly by the model." (Kashyap, 2) **Does not include the wrong predictions**
Recall - "Number of apples predicted correctly by the model/Total number of apples."(Kashyap, 3) **Includes the wrong predictions.**

When training a model and viewing how accurate predictions are made, we assume that accuracy is a good predictor of how good the model predicts. However, that is deceiving, since there are other measurements like recall and precision. The importance of these two metrics depends on the data you are using and what you are trying to predict. We want a **HIGH** recall when it comes to healthcare like detecting cancer. Let's assume, you were screened for cancer, the model predicts you have cancer, you get called in to do extra screenings. However, the extra screenings show that you **did not** have cancer. The model was wrong, because it had a higher recall. Great, good news you are healthy! Now, same example but you have a higher presion and lower recall, you get screened for cancer and the model **does not** detect cancer. Years later you get a normal checkup and you find out that you have stage 2 cancer. The model was wrong, you did have cancer. (Kashyap, 4)

Article that was used for this scenario:
https://towardsdatascience.com/precision-and-recall-a-simplified-view-bc25978d81e6

Back to Problem

We just discussed precision and recall. Since K nearest neighbor had the highest accuracy, we will dive deeper into that's classification report and discuss recall and precision for this classifier

- **K Nearest Neighbors:**
    - **Recall**
        - **No:** 98%
        - **Yes:** 13%
    - **Precision**
        - **No:** 92%
        - **Yes:** 34%

Like we stated in the above section, when it comes to healthcare we would rather care about recall then precision when it comes to training the model. We can see that **No** had a 98% recall rate and **Yes** had a 13% recall rate. “Yes” looks low right? That is due to the fact that we only could use a limited number of data rows to train this model (60,000) because anything higher would take too long to train and with the software I was using it wouldn't allow it to run for that long. With the 60,000 rows only around 5,000 said **Yes** to having heart disease and the rest were **No’s.** I am confident that the recall for **Yes** would go higher if I was able to use all 300,000+ rows of data.

Unsupervised Learning Portion

In this portion of the project I looked at a different dataset that consisted of indiviudals and if they have diabetes. I used Unsupervised learning to cluster this dataset.

Overview of Algorithms Used and Explained + Pseudo Code/Real Code

I used the library sklearn.cluster that is used to create clusters in the scatter plot. The features that are included in this dataset were:

- Pregnancies
- Glucose
- Bloodpressre
- SkinThickness
- Insulin
- BMI
- DiabetesPedigree Function
- Age
- Outcome (1 = diabetes or 0 = No diabetes)

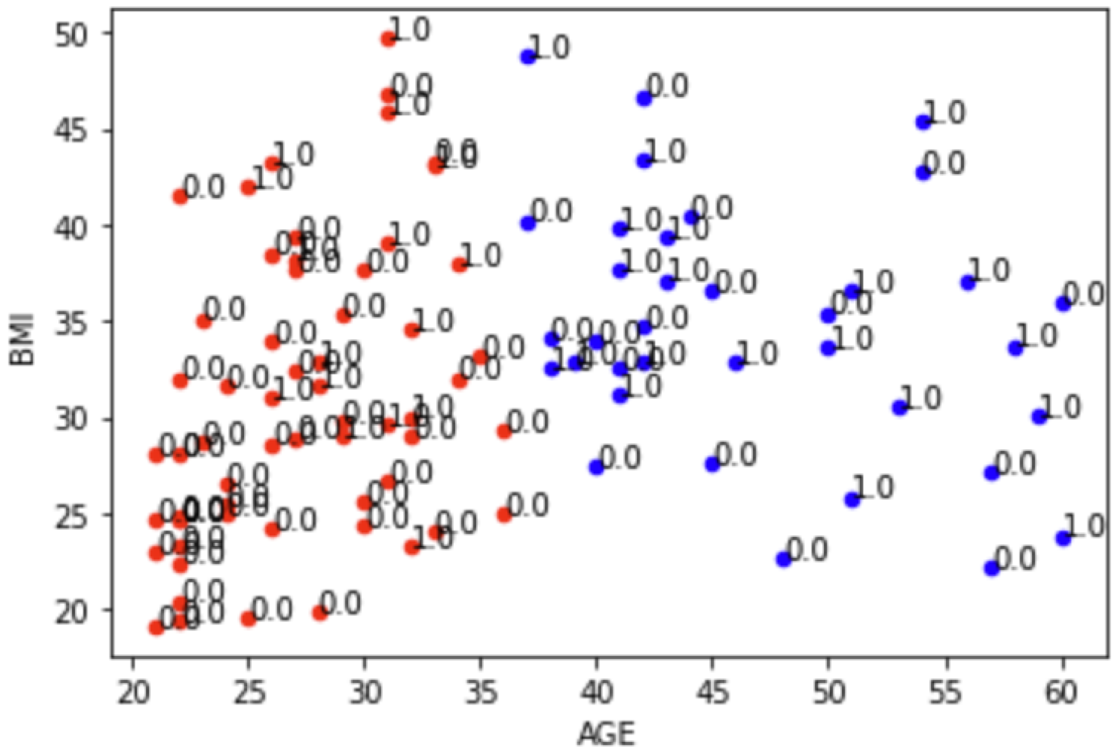
However, I didn’t want to use all the features and just use BMI and Age since there is relation to both of those features. I clustered the data into 2 clusters (K) and fit the model with Kmeans and used a for loop to assign the color and labels to the data points and to show the clusters.
Pseudo code:
Set x as Rows BMI and Age

Set the number of clusters you want
Use Kmeans
Fit the classification
Iterate each row and set the cluster with a label and color
Show scatter plot

Real Code:

```
x = diabetes_df[["BMI","Age"]]
k = 2
clt = KMeans(n_clusters=k)
clt.fit(x)
cluster_lables = clt.predict(x)
colors = ["b","r"]
plt.figure(figsize=(16,10))
for i, row in diabetes_df.iterrows():
 curr_label = cluster_lables[i]
 curr_color = colors[curr_label]
 plt.scatter(row["Age"],row["BMI"],c=curr_color, s=20)
 plt.text(row["Age"],row["BMI"],row["Outcome"],size=10)
plt.xlabel("AGE")
plt.ylabel("BMI")
plt.show()
```

Analysis/Summary



Disclaimer: the original dataset had WAY more rows of data, but I cut it down due to the scatter plot looking too messy, but this does not mess up the results. Anway, this was the graph I got. We can see the 2 clusters I stated in the above section. I wanted to use 2 clusters because we can see who has diabetes and who doesn't. We can see that red tends to have less individuals that have diabetes, AND they are at a lower BMI and a younger age. We can see the blue cluster has more individuals that have diabetes and they tend to be older with high BMIs. This data seems to look correct because some of the main factors of diabetes is Age and BMI. According to the CDC, "You're at risk for type 2 diabetes if you: are overweight, are 45 years or older"(CDC, 4). Having a combination of those two factors gives you a higher chance to have diabetes. However, we can see some strange datapoints, but that can be caused by either the individual as type 1 diabetes, despite being healthy and young.

Sources:

https://careerfoundry.com/en/blog/data-analytics/what-is-a-decision-tree/#:~:text=Decision%20trees%20are%20extremely%20useful,%2C%20data%20classification%2C%20and%20regression.

https://towardsdatascience.com/knn-algorithm-what-when-why-how-41405c16c36f

https://www.upgrad.com/blog/naive-bayes-explained/#:~:text=Naive%20Bayes%20is%20suitable%20for,input%20variables%20than%20numerical%20variables.

https://towardsdatascience.com/machine-learning-project-17-compare-classification-algorithms-87cb50e1cb60

https://www.cdc.gov/diabetes/basics/risk-factors.html

https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease (Heart Disease Dataset)

https://www.kaggle.com/datasets/akshaydattatraykhare/diabetes-dataset (Diabetes Dataset)