Cancer Prediction

Dataset

- Using the Wisconsin Breast Cancer dataset
  - Data retrieved from Kaggle
- Data computed from a digitized image of a fine needle aspirate (FNA) of a breast mass.
  - Each row represents one image
  - Each column is one feature of a cell nuclei

Features

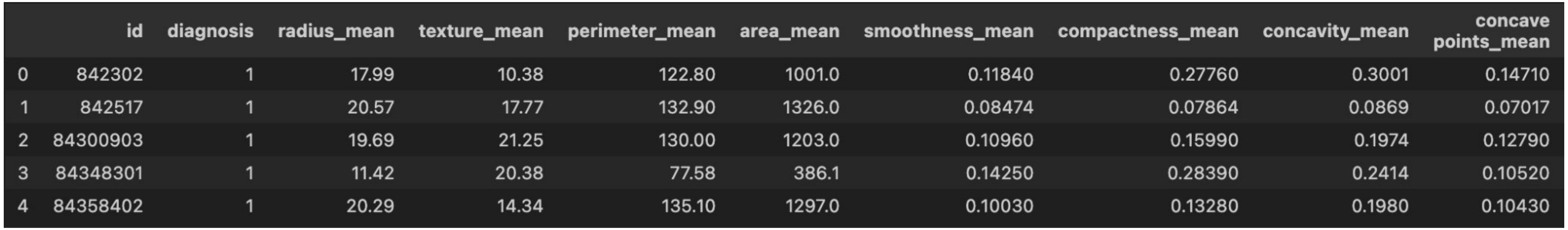
a. radius (mean of distances from center to points on the perimeter)
b. texture (standard deviation of gray-scale values)
c. perimeter
d. area
e. smoothness (local variation in radius lengths)
f. compactness (perimeter^2 / area - 1.0)
g. concavity (severity of concave portions of the contour)
h. concave points (number of concave portions of the contour)
i. symmetry
j. fractal dimension ("coastline approximation" - 1)

For each feature, there are three statistics:

- mean
- se (standard error)
- worst (largest)

The remaining columns are the id and the diagnosis

- Diagnosis is either '1' (malignant) or '0' (benign)



	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean
0	842302	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710
1	842517	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017
2	84300903	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790
3	84348301	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520
4	84358402	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430

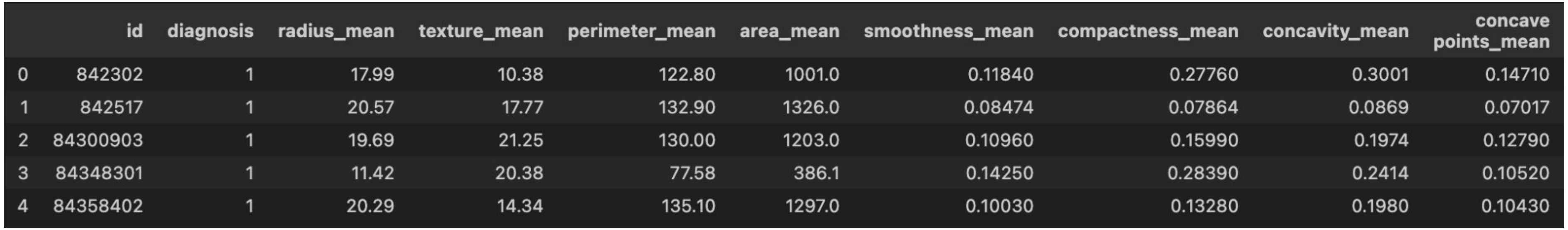
To see all rows and columns (can't show all due to slide size), visit Jupyter notebook

Separating Features and Labels

- Need to separate data into features and labels
- Model will use features to predict labels
- Features (listed previously):
  - [radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, fractal dimension]
  - Also includes their three different versions (mean, standard error, largest)
- Label: Diagnosis
  - Diagnosis is either '1' (malignant) or '0' (benign)
- Store features and labels into separate dataframes (X and y respectively)

Label

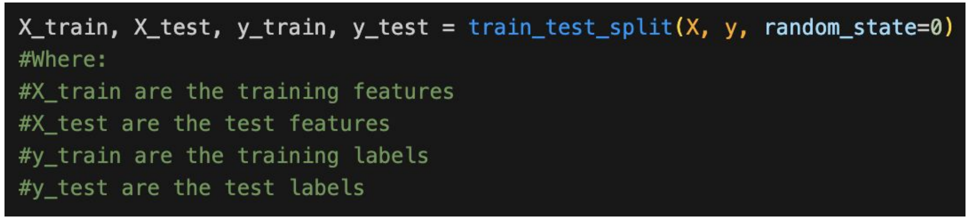
Features



	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean
0	842302	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710
1	842517	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017
2	84300903	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790
3	84348301	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520
4	84358402	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430

Splitting into Training and Testing Sets

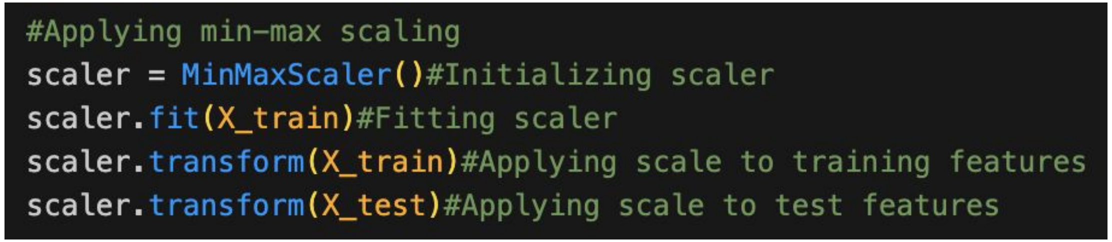
- Because we lack a test dataset, have to split current dataframe into a training and test dataset
  - Training dataframe is 75% of original dataframe
  - Test dataframe is 25% of original dataframe
- Splits current 2 dataframes (X, y) into 4:
  - Training
    - X_train (training features)
    - Y_train (training labels)
  - Test
    - X_test (test features)
    - y_test (test labels)
- Use test dataframe accuracy to evaluate model



```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
#Where:
#X_train are the training features
#X_test are the test features
#y_train are the training labels
#y_test are the test labels
```

Min-Max Scaling

- For some algorithms, it’s important that features all on same scale
  - Otherwise some features will be treated as less important by model
- Use min-max scaling to convert features to same scale
  - Makes all feature data on scale of 0-1 relative to their respective min and max values
  - There are other options to make similar scales



```
#Applying min-max scaling
scaler = MinMaxScaler()#Initializing scaler
scaler.fit(X_train)#Fitting scaler
scaler.transform(X_train)#Applying scale to training features
scaler.transform(X_test)#Applying scale to test features
```

Picking a Model

Regression vs Classification

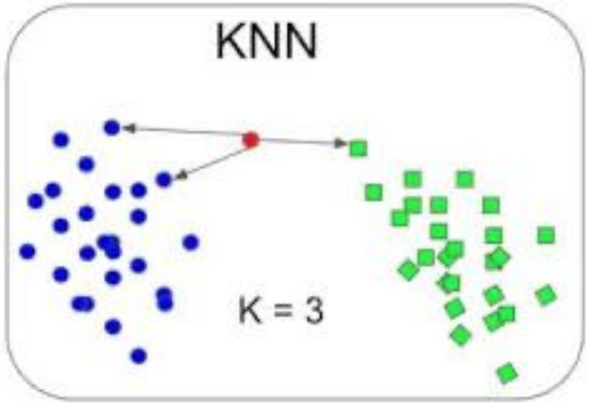
- Regression models predict numerical, continuous values
- Classification models predict categorical values

Picking models

- Our labels are categorical → classification problem
    - Specifically binary classification
- There's many classification models → choose the one that gets highest test dataset accuracy
- We'll try 2 different classifier models:
    - k-nearest neighbor
    - logistic regression

K-Nearest Neighbors

- Looks for ‘k’ amount of closest data points (in the image, 3) to the datapoint it’s trying to predict
  - We’ll be using 1 for our model
- Assigns most common class of the neighbor data points to the unknown datapoint
- Accuracy: 0.94



In this example, the unknown point would be classified as the blue class since that’s the most common class of the 3 neighbors

Logistic Regression

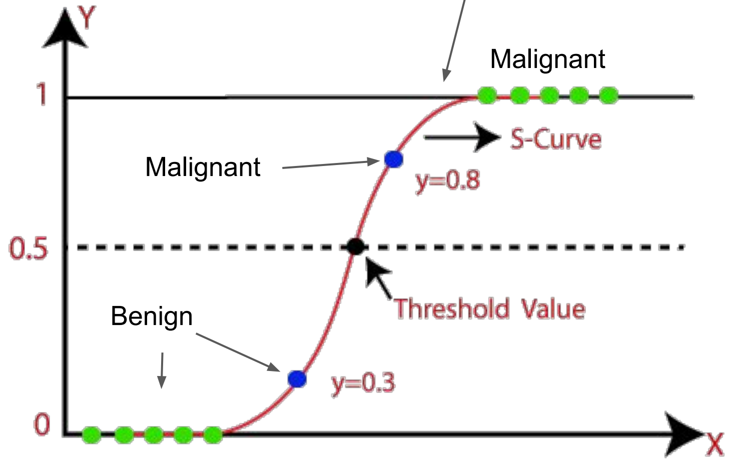
- Produces an s-curve with max 0 and min 1
- Assigns coefficients (weights) to each feature (radius, etc.)
- Adds resulting numbers plus a bias (b-hat)
- Inputs resulting number (y-hat) into logistic regression equation
- If resulting number f(x) is above 0.5, assigned class of 1 (malignant)
- If below 0.5, assigned class of 0 (benign)
- Accuracy: 0.96

Combine features and weights (w=weight, x=feature value)

$$\hat{y} = \hat{b} + \hat{w}_1 \cdot x_1 + \cdots \hat{w}_n \cdot x_n$$

Input combined sum into logistic regression formula

$$f(x) = \frac{1}{1 + e^{-x}}$$



Values ≥ 0.5 = Class 1
Values< 0.5 = Class 2

Conclusion

- Since the logistic regression model had a higher accuracy, it is better
  - Albeit only slightly better; could be due to luck. Essentially splitting hairs

General process of making a model

1. Find dataset
2. Split dataset into labels and features
3. Split further into training and test datasets
4. Pick a model
5. Train model
6. Evaluate via test dataset