Breast Cancer Diagnosis

Using Data Science to predict tumor malignancy based on nucleus attributes identified via digitized images of breast tissue extracted via fine needle aspiration

Carolyn Nohejl

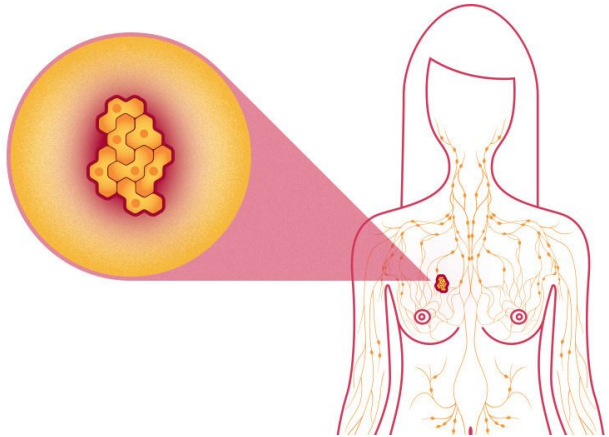
September 2017

Agenda

1. Problem Statement and Background
2. Dataset and Analysis
3. Modeling Approach and Results
4. Conclusion
5. Next Steps
6. Appendix

Jupyter Notebooks:

- Exploratory Data Analysis
- Modeling



Project Problem

- **Background:** Breast cancer is a threatening illness, and early diagnosis leads to greater chances of survival. It is known that the size and shape of cancer cell nuclei tend to be abnormal.[1]

- **Problem statement:** Predict the malignancy of a tumor based on cell nucleus attributes identified via digitized images of breast tissue extracted via fine needle aspiration via the Diagnostic Wisconsin Breast Cancer Database.
  - *Problem and dataset leveraged from Kaggle.*

- **Machine learning problem:** This is a supervised learning classification problem where I will be predicting a binary variable - malignant or benign, based on a series of continuous variables.

[1] **What do doctors look for in biopsy and cytology specimens?**
https://www.cancer.org/treatment/understanding-your-diagnosis/tests/testing-biopsy-and-cytology-specimens-for-cancer/what-doctors-look-for.html. 9/4/2017

Potential Impact and Hypothesis

- **Potential impact:** The algorithm could be leveraged to diagnose whether a tumor is malignant or benign leveraging a minimally invasive method of extracting breast tissue. An accurate minimally invasive method will facilitate early screening which has been shown to increase the rate of survival.

- **Success looks like:** Define achievable confidence rate of predicting if a tumor is malignant or benign using a minimally invasive method of tissue extraction and computer vision diagnostic systems to extract nucleus features.

- **Hypothesis:** I expect perimeter to have the most impact in predicting if the tumor is malignant or benign given that it is a reflection of nucleus shape and size.

Dataset

Cell nucleus features are computed from a digitized image of a fine needle aspirate of a breast mass.[2]

- **Samples:** Dataset includes 569 samples
- **Class distribution:** Classified as malignant (M) or benign (B) with decent class distribution 212:357 = 37%/63%.
- **Features:**
  - ID number, diagnosis classifier (M= malignant, B= benign), and 10 features are computed for each cell nucleus leveraging a digital scan.
  - The mean, standard error and "worst" (mean of the three largest values) for each feature were computed, resulting in a total of 30 features.
  - All features were numerically modeled such that larger values will typically indicate a higher likelihood of malignancy

[2]*Dataset: https://www.kaggle.com/uciml/breast-cancer-wisconsin-data*

Dataset

Type	Feature	Feature Description
Size	Radius Area	Mean of distances from center to points on the perimeter Number of pixels in the interior and half of the pixels on the perimeter
Size and Shape	Perimeter	Total distance between points on the perimeter
Shape	Smoothness Compactness Concavity Concave points Symmetry Fractal dimension	Local variation in radius lengths Perimeter^2 / area - 1.0 Severity of concave portions of the contour Number of concave portions of the contour Length difference between lines perpendicular to the major axis to the cell boundary "coastline approximation" - 1
Texture	Texture	Standard deviation of gray-scale values

Data Cleaning

- **Data Quality:**
  - No missing attribute values - very clean
  - Manually identified “0”s in the data. They were only found in the concavity and concave points columns (13 of them) for benign cells. I left these cases in as healthy cells are known to be round, therefore it makes sense that concavity could be 0, rather than deleting based on the assumption that the measurement was not taken.
  - **Summary:** I did not remove any samples

- **Train/test split:** Broke the data into training and test data frames (70%/30% split)
  - df_train = 398 samples (70%)
  - df_test = 171 samples (30%)

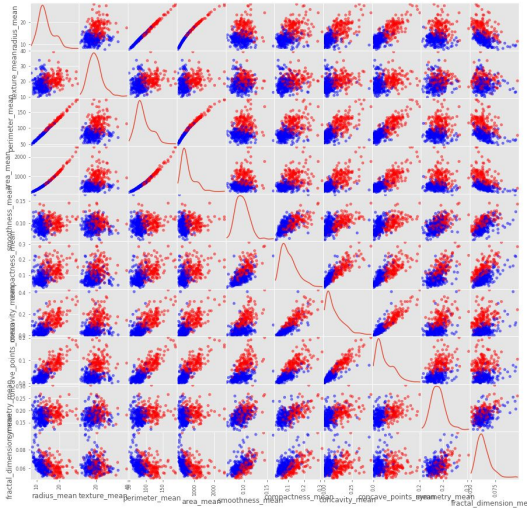
Exploratory Analysis

- **Plotting:**
  - Visualized data via a scatter plot to confirm that features were modeled such that higher values aligned with malignancy (see next slide) - confirmed for “mean” and “worst” data
  - Visualized the correlation amongst the attributes → as expected, correlation is very high across the “means” and “worst” data.
- **Outliers:**
  - Broke the training dataframe into two new dataframes by classifier and assessed the number of outliers
    - Benign: of 249 samples, 111 outliers
    - Malignant: of 149 samples, 52 outliers
  - **Conclusion:** Outliers represent >30% of the dataset. Given a large number of variables, an outlier for one variable may be important for another. Therefore I did NOT remove outliers.

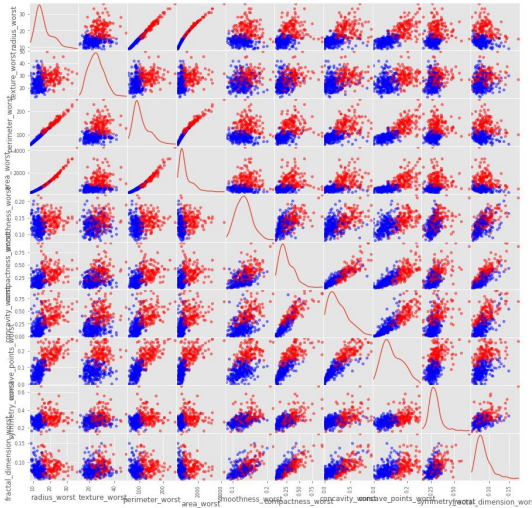
Scatter Plots

Confirmed authors' statement that in general features were modeled such that higher values aligned with malignancy (red).

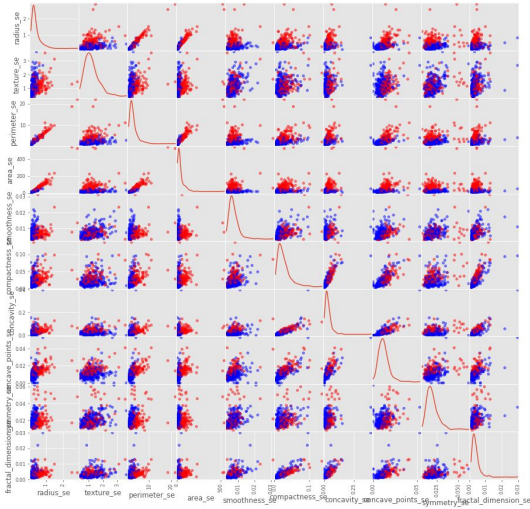
- This generally seemed true for the means and worst data, with the exception of smoothness, symmetry, fractal dimension, and texture. Therefore I hypothesize that these features will be less valuable in the models.
- For standard error, the he relationship between higher values and malignancy is less clear
- See appendix for larger plots



Mean

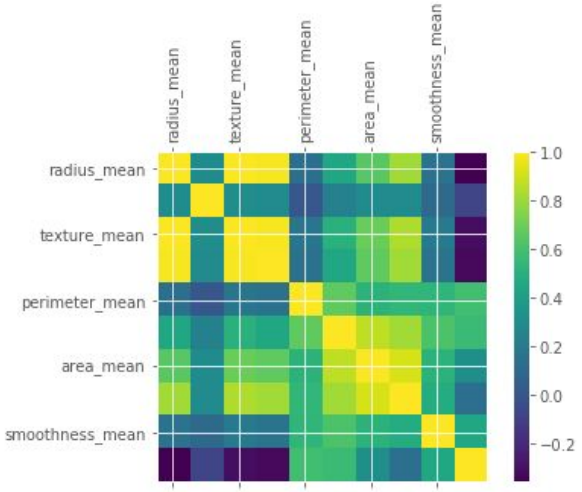


Worst

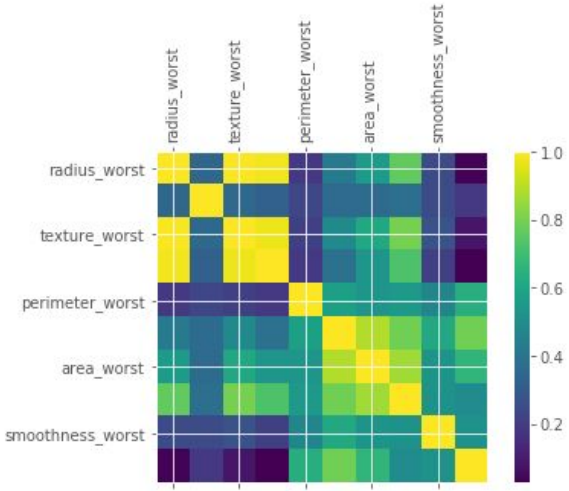


Standard Error

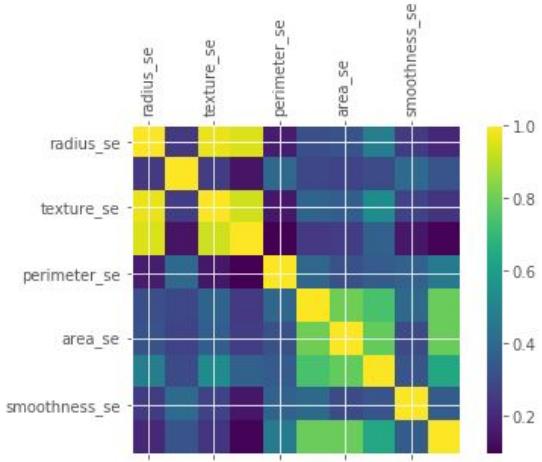
Plotting - Correlation Heatmap



Mean



Worst



Standard Error

Mean seems to be the most highly correlated, followed by worst, then SE. This is not surprising given that the features were numerically modeled such that larger values will typically indicate a higher likelihood of malignancy. This may present challenges when using Logistic Regression.

Modeling Approach

Leveraged the following modeling techniques and compared accuracy using 10 fold cross validation:

1. **Random Forest:** used random forest on all 30 features to identify the top features. Used this first given that it is less sensitive to multicollinearity.

2. **Logistic Regression** (statsmodels and scikit learn)**:**
   a. Performed backward selection method leveraging top 10 features from random forest model
   b. Performed forward selection method starting top feature from random forest model

3. **K Nearest Neighbors**
   a. Performed KNN on top 3 features from Random Forest model, and top features from the logistic regression forward selection model.

1. Random Forest

Approach:

- Input all 30 features into the model, with the classifier (removed ID)
- N_estimators = 1000

Results:

- OOB score: 95% accuracy
- 10-fold cross validation on the training set: 95% accuracy
- **10-fold cross validation on the entire data set: <u>94% accuracy</u>**

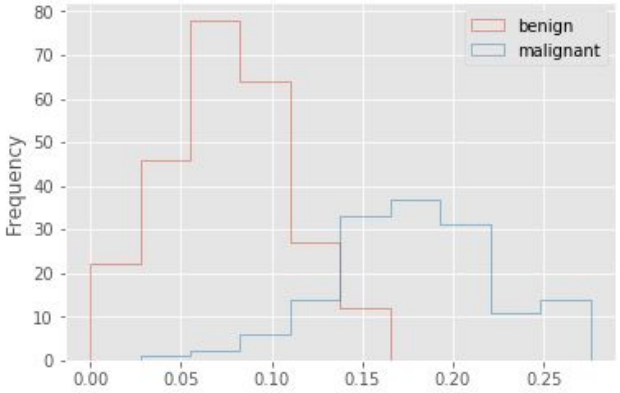
Discussion:

- Only half of the feature types were represented and included shape and area measures (concave points, perimeter, radius, area, concavity). Texture was not as relevant.
- **I will use 3 of the four top features in the KNN model later:**
  - **Concave points worst, perimeter worst, and radius worst**
  - Skipped concave points mean as I expected it to be highly correlated with concave points worst

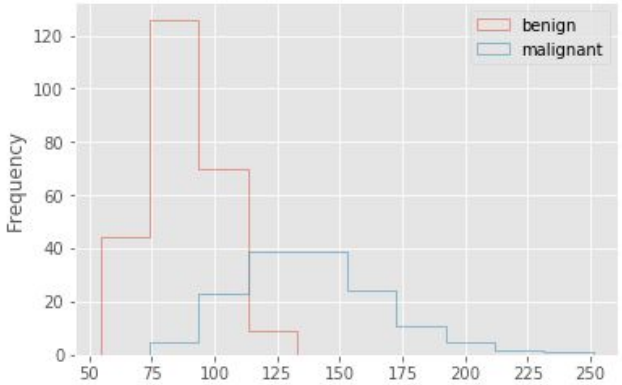
```
In [22]: sorted(zip(model.feature_importances_, \
                    X.columns.values), reverse = True)

Out[22]: [(0.16715931454388719, 'concave points_worst'),
          (0.13366127363499894, 'concave points_mean'),
          (0.1174542576640398, 'perimeter_worst'),
          (0.099679487882633755, 'radius_worst'),
          (0.093355900316311788, 'area_worst'),
          (0.081154916673909866, 'concavity_mean'),
          (0.054142055938401851, 'concavity_worst'),
          (0.049025239156765835, 'perimeter_mean'),
          (0.035210498515225508, 'radius_mean'),
          (0.032636132921040235, 'area_mean'),
          (0.022275230670363317, 'area_se'),
          (0.018244777522621172, 'compactness_worst'),
          (0.011895227652925895, 'compactness_mean'),
          (0.01176052704479969, 'radius_se'),
          (0.011495530457791623, 'texture_worst'),
          (0.011005383867656664, 'symmetry_worst'),
          (0.0074339374683463234, 'perimeter_se'),
          (0.0072703222121763503, 'smoothness_worst'),
          (0.0062425646185192123, 'texture_mean'),
          (0.0059140307436010477, 'fractal_dimension_worst'),
          (0.0040081706736959022, 'concavity_se'),
          (0.0028348740774689174, 'concave points_se'),
          (0.0025285891563906118, 'smoothness_mean'),
          (0.0024003392763913998, 'compactness_se'),
          (0.0022897662010329203, 'fractal_dimension_mean'),
          (0.0022015678487127014, 'smoothness_se'),
          (0.0020233603969543861, 'symmetry_mean'),
          (0.0018211660219076098, 'fractal_dimension_se'),
          (0.0015716804964940427, 'symmetry_se'),
          (0.0013038763449350079, 'texture_se')]
```

1. Random Forest - Top Features

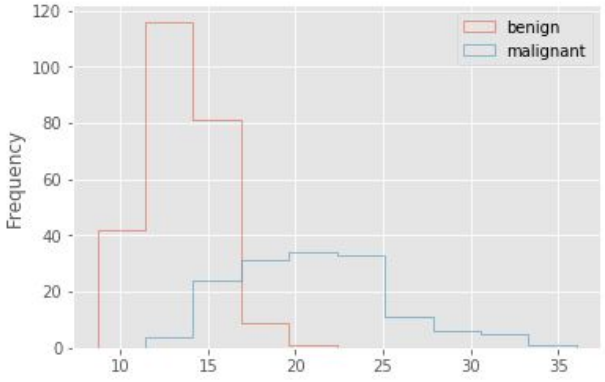
Not surprisingly, the top 3 features show low overlap between the benign and malignant data subsets.



Concave points worst



Perimeter worst



Radius worst

Included in EDA workbook

2. Logistic Regression - Backward Selection

Approach #1: Statsmodels - Backward Selection Method

- Input features contributing >5% to the random forest model (concave points worst, concave points mean, perimeter worst, radius worst, area worst, concavity mean, concavity worst)
- Removed features one by one based on their p-values

Results:

- Stopped removing features when all p-values were significant, resulting in 5 top features: concave points mean, perimeter worst, area worst, concavity mean, concavity worst
- Pseudo R-squ.: 0.8149

Scikit learn:

- Logistic regression: 92% accuracy on test data (within training set)
- 10 fold cross validation on the training set: 91% accuracy
- **10 fold cross validation on the entire data set: 92% accuracy**

Discussion: I'm surprised that the results from the backward selection were as good as they were - I would have expected multicollinearity to be more of an issue.

```
result.summary()
```

Logit Regression Results

Dep. Variable:	diagnosis	**No. Observations:**	398
Model:	Logit	**Df Residuals:**	392
Method:	MLE	**Df Model:**	5
Date:	Sat, 16 Sep 2017	**Pseudo R-squ.:**	0.8149
Time:	12:46:18	**Log-Likelihood:**	-48.724
converged:	True	**LL-Null:**	-263.17
		LLR p-value:	1.745e-90

	coef	std err	z	P>|z|	[95.0% Conf. Int.]
const	-2.5486	3.930	-0.648	0.517	-10.251 5.154
concave_points_mean	115.5608	26.044	4.437	0.000	64.516 166.606
perimeter_worst	-0.2142	0.084	-2.562	0.010	-0.378 -0.050
area_worst	0.0219	0.006	3.695	0.000	0.010 0.033
concavity_mean	-38.2359	12.304	-3.108	0.002	-62.351 -14.121
concavity_worst	16.8824	3.911	4.317	0.000	9.218 24.547

2. Logistic Regression - Forward Selection

Approach #2: Statsmodels - Forward Selection Method

- Started with the top feature from the random forest model, concave points worst
- Added features one by one based on their p-values (skipping concave points mean)

Results:

- Final feature set is: concave points worst and perimeter worst
- Pseudo R-squ.: .7754

Scikit learn:

- Logistic regression on the training set: 91% accuracy
- 10 fold cross validation on the training set: 91% accuracy
- **10 fold cross validation on the entire data set: 92% accuracy**

```
result.summary()
```

Logit Regression Results

Dep. Variable:	diagnosis	**No. Observations:**	398
Model:	Logit	**Df Residuals:**	395
Method:	MLE	**Df Model:**	2
Date:	Sat, 16 Sep 2017	**Pseudo R-squ.:**	0.7754
Time:	13:54:07	**Log-Likelihood:**	-59.119
converged:	True	**LL-Null:**	-263.17
		LLR p-value:	2.400e-89

	coef	std err	z	P>|z|	[95.0% Conf. Int.]
const	-18.1385	2.430	-7.466	0.000	-22.900 -13.377
concave_points_worst	41.2781	7.590	5.438	0.000	26.401 56.155
perimeter_worst	0.1160	0.020	5.693	0.000	0.076 0.156

Discussion:

- I was surprised to see that the backward and forward selection results were similar - I would have expected the forward selection results to be better given that I used less features, reducing multicollinearity.
- **I will leverage concave points worst and perimeter worst in my KNN model given that they were the top features from the forward selection method.**

3. K Nearest Neighbors (KNN)

Approach:

- Created 2 KNN models:
  a. Leveraging top 3 features from random forest: concave points worst, perimeter worst, and radius worst
  b. Leveraging top features from logistic regression, forward selection method (2 features): concave points worst, perimeter worst
- 50/50 train test fit
- 5 fold CV on training data, identified best number of neighbors = 5 for both models
- 10 fold CV

Results:

- **10 fold CV leveraging top 3 features from random forest: 95% accuracy**
- 10 fold CV leveraging top features from logistic regression, addition method (2 features): 94% accuracy

Conclusion

The KNN model incorporating concave points worst, perimeter worst, and radius worst features provided the highest **10 fold cross validation accuracy of 95%**, to predict if a tumor is malignant or benign.

Next Steps

- Perform feature engineering and adjust model parameters in an effort to optimize the results.
- Leverage Lasso technique to identify other combinations of top features.
- Consider removing outliers from top features used for the models.
- Compare my work and leverage ideas from others on Kaggle. Aim to increase my accuracy beyond that Street, Wolberg, and Mangasarian[3], whom achieved 97% accuracy with their model.
- After optimizing the model(s) and identifying the achievable confidence rate, research the accuracy rates of other biopsy and analysis methods to understand commercial viability of fine needle aspiration and nucleus feature definition using a computer vision diagnostic system. It will be important to understand not only the accuracy of existing methods, but also the burden of each of these techniques to the patient and provider, and also the cost to the healthcare system.

[3]*Nuclear Feature Extraction for Breast Tumor Diagnosis, Street, et al, 1992.*

Appendix

Approach: Part 1

1

Initial clean up:
- Changed M,B → 0,1 (2 approaches
- Changed column names

Full Data Set (569 samples):
EDA - examined outliers

Train (df_train - 398 samples) = 70% of df | Test (df_test - 171 samples) = 30% of df

2

Random Forest with 30 features using the training set → select top features

Train - 60% | Test - 40%

3

Random Forest with top features
- 10 fold cross validation on training set, then full dataset

Train - 60% | Test - 40%

Random Forest w/top features- 10 fold cross validation on training set

Top 3 features to KNN →

Random Forest with top features - 10 fold cross validation on full dataset

4

Logistic Regression - statsmodels
- Top features from Random Forest + constant (reduction method)
- Start with best feature from Random forest + constant and add features

Used training set

Used training set

5

Logistic Regressions - scikit learn
- Perform with features from the addition method on the **training set**
- Perform with features from the addition method on the **entire set**
- Performed with all 30 features

Top 2 features to KNN →

Train = 60% of df_train | Test = 40% of df_train

Logistic Regression - 10 fold cross validation

Logistic Regression - 10 fold cross validation

Approach: Part 2

Full data set (569 samples)

Train (df_train - 398 samples) = 70% of df | Test (df_test - 171 samples) = 30% of df

6

Run two KNN models:

- Top 3 features from Random Forest
- Top 2 features Logistic Regression Addition method

Train - 50% | Test - 50%

Train - 50% | Test - 50%

Run 10 fold cross validation on **the two KNN models** first on the training set, then on the entire dataset

KNN - 10 fold cross validation

KNN - 10 fold cross validation

7

Select the best model

DataFrames - EDA

DataFrame Name	Description
df	Entire dataset, 569 samples
df_train	Training set, 70% of entire dataset = 398 samples, randomly selected (42)
df_test	Test set, 30% of entire dataset = 171 samples, randomly selected (42)
df_train_means	Subset of df_train with means features with diagnosis
corr_train_means	Correlation matrix for df_train_means
df_train_worst	Subset of df_train with worst features with diagnosis
corr_train_worst	Correlation matrix for df_train_worst
df_train_se	Subset of df_train with se features with diagnosis
corr_train_se	Correlation matrix for df_train_se
df_train_benign	Subset of training data (df_train) with benign diagnosis (249 samples)
df_train_malignant	Subset of training data (df_train) with malignant diagnosis (149 samples)
df_train_benign_outlier	Replicated training data benign dataframe for outlier assessment
df_train_malignant_outlier	Replicated training data malignant dataframe for outlier assessment

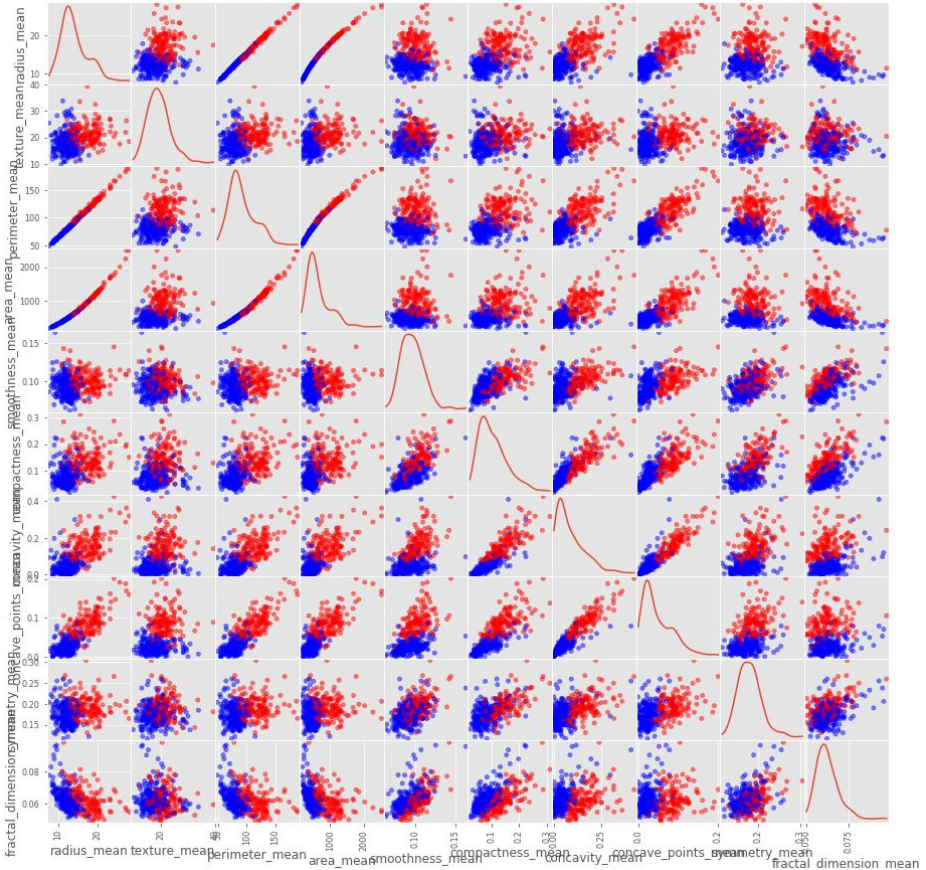
DataFrames - Modeling

DataFrame Name	Description
X	df_train without id and diagnosis - used for Random Forest, Logistic Regression
y	df_train with classifier only - df_train.diagnosis - used for Random Forest, Logistic Regression
train_X, test_X, train_y, test_y	60/40 split of X and y above - for Random Forest
X_final	df (entire dataset) without id and diagnosis - for Random Forest, Logistic Regression
y_final	df (entire dataset) with classifier only - df.diagnosis - used for Random Forest, Logistic Regression
train_X_LR	df_train including top 7 variables from Random Forest, removed features during modeling - for Logistic Regression Backward Selection
train_c_LR	df_train including diagnosis only - used for Logistic Regression Backward Selection
train_X_LR2	df_train including concave points only, added features during modeling - for Logistic Regression Forward Selection
trian_c_LR2	df_train including diagnosis only - for Logistic Regression Forward Selection
W	df_train including only the 5 top features from Logistic Regression Backward Selection - for SCIKIT learn
c	df_train including diagnosis only - for SCIKIT learn Logistic Regression for Backward Selection
W_train, W_test, c_train, c_tes	60/40 split for W and c above - or SCIKIT learn Logistic Regression for Backward Selection
W_final	df (entire dataset) w/only the 5 top features from Logistic Regression Backward Selection - for SCIKIT learn Logistic Regression for Backward Selection
c_final	df (entire dataset) including diagnosis only - for SCIKIT learn Logistic Regression for Backward Selection
P	df (entire dataset) w/only the 2 top features from Logistic Regression Forward Selection - for SCIKIT learn Logistic Regression for Forward Selection
r	df_train including diagnosis only - for SCIKIT learn Logistic Regression for Forward Selection
P_final	df (entire dataset) w/only the 2 top features from Logistic Regression Forward Selection - for SCIKIT learn Logistic Regression for Forward Selection
r_final	df (entire dataset) including diagnosis only - for SCIKIT learn Logistic Regression for Forward Selection
df_knn_postrf	df_train with only top 3 features from Random Forest and classifier - for KNN
XX	df_knn_postrf features only, scaled - for KNN
cc	df_knn_postrf classifier only - for KNN
X	df with XX columns - top 3 features from Random forest - for KNN
c	df with classifier only - for KNN
df_knn_LR	df_train with only top 2 features from Logistic Regression and classifier - for KNN
LR	df_knn_LR features only, scaled - for KNN
l (lowercase L)	df_knn_LR classifier only - for KNN
R	df with LR columns - top 2 features from Logistic Regression - for KNN
q	df with classifier only - for KNN

Scatter Plots - Means

Confirmed authors' statement that in general features were modeled such that higher values aligned with malignancy (red).

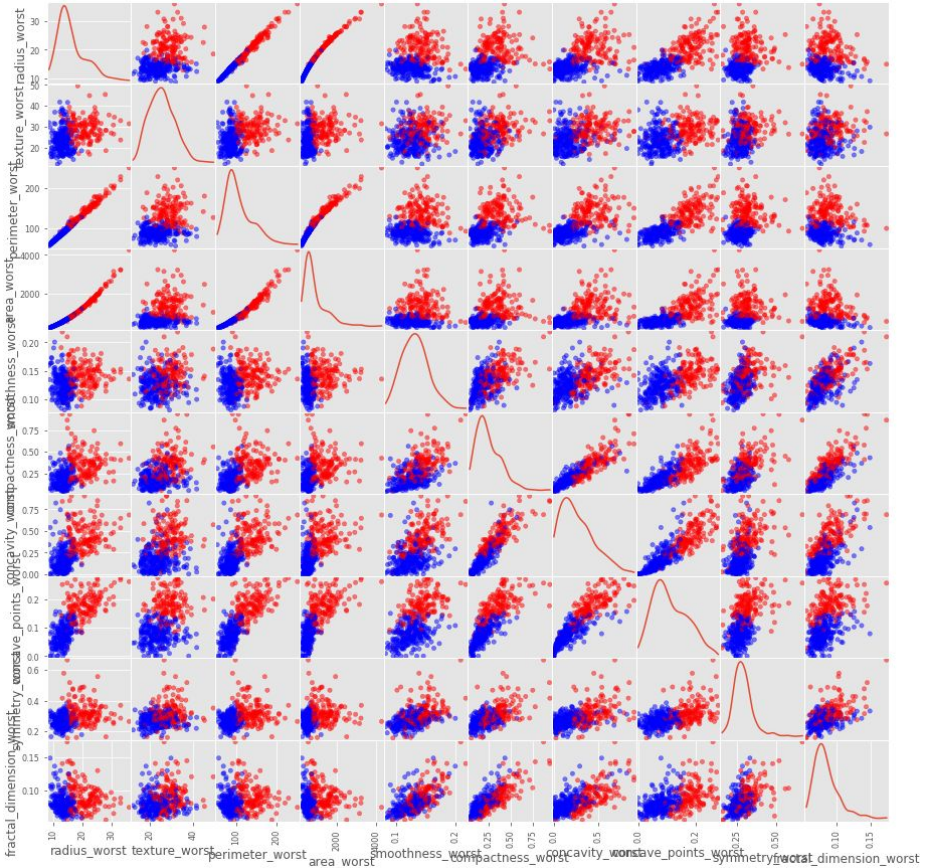
This generally seemed true for the means data, **with the exception of smoothness, symmetry, fractal dimension, and texture.** Therefore I hypothesize that these features will be less valuable in the models.



Scatter Plots - Worst

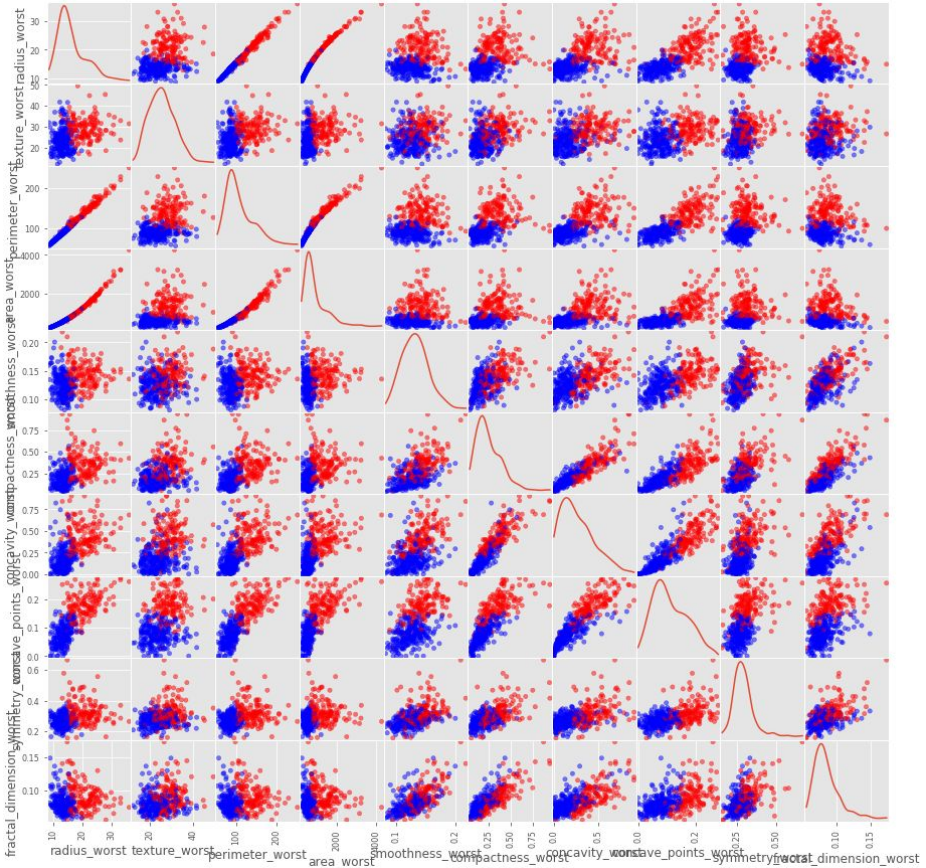
Confirmed authors' statement that in general features were modeled such that higher values aligned with malignancy (red).

This generally seemed true for the worst data, **with the exception of smoothness, symmetry, fractal dimension, and texture.** Therefore I hypothesize that these features will be less valuable in the models.



Scatter Plots - Standard Error

For standard error, the he relationship between higher values and malignancy is less clear



Outliers - Benign

Of 249 samples, 111 of them are outliers beyond the 1st and 3rd quartiles, leaving only 138 samples.

```
for col_name in df_train_benign_outlier.columns:
    if col_name in ['id', 'diagnosis']:
        continue

    Q1 = df_train_benign_outlier[col_name].quantile(0.25)
    Q3 = df_train_benign_outlier[col_name].quantile(0.75)
    IQR = Q3 - Q1
    df_train_benign_outlier = df_train_benign_outlier[(df_train_benign_outlier[col_name] \
            > (Q1 - 1.5 * IQR)) & (df_train_benign_outlier[col_name] < (Q3 + 1.5 * IQR))]
    print(col_name, Q1, Q3, df_train_benign_outlier.shape)
```

```
('radius_mean', 11.22, 13.38, (246, 32))
('texture_mean', 15.1875, 19.784999999999997, (234, 32))
('perimeter_mean', 71.3475, 86.07, (234, 32))
('area_mean', 381.375, 553.05, (232, 32))
('smoothness_mean', 0.083705, 0.1007, (231, 32))
('compactness_mean', 0.05623500000000001, 0.097875, (224, 32))
('concavity_mean', 0.020415, 0.058492499999999996, (219, 32))
('concave_points_mean', 0.014865, 0.030934999999999997, (212, 32))
('symmetry_mean', 0.1555, 0.186275, (209, 32))
('fractal_dimension_mean', 0.05853, 0.06562, (205, 32))
('radius_se', 0.21, 0.3278, (200, 32))
('texture_se', 0.7655749999999999, 1.3507500000000001, (194, 32))
('perimeter_se', 1.43575, 2.225, (192, 32))
('area_se', 14.852500000000001, 23.6225, (188, 32))
('smoothness_se', 0.0049015, 0.007831, (184, 32))
('compactness_se', 0.01053, 0.021825000000000004, (174, 32))
('concavity_se', 0.010135, 0.0233475, (163, 32))
('concave_points_se', 0.00583, 0.0099215, (161, 32))
('symmetry_se', 0.01494, 0.021519999999999997, (155, 32))
('fractal_dimension_se', 0.001914, 0.0033275, (152, 32))
('radius_worst', 12.81, 14.91, (149, 32))
('texture_worst', 19.64, 25.72, (148, 32))
('perimeter_worst', 82.91, 97.2875, (147, 32))
('area_worst', 511.5, 685.55, (146, 32))
('smoothness_worst', 0.11105, 0.137275, (146, 32))
('compactness_worst', 0.109675, 0.203025, (143, 32))
('concavity_worst', 0.063345, 0.18775, (141, 32))
('concave_points_worst', 0.051039999999999995, 0.0866, (139, 32))
('symmetry_worst', 0.24355, 0.29800000000000004, (138, 32))
('fractal_dimension_worst', 0.0689, 0.0816075, (138, 32))
```

Outliers - Malignant

Of 149 samples, 52 of them are outliers beyond the 1st and 3rd quartiles, leaving only 97 samples.

```
for col_name in df_train_malignant_outlier.columns:
    if col_name in ['id', 'diagnosis']:
        continue

    Q1 = df_train_malignant_outlier[col_name].quantile(0.25)
    Q3 = df_train_malignant_outlier[col_name].quantile(0.75)
    IQR = Q3 - Q1
    df_train_malignant_outlier = df_train_malignant_outlier[(df_train_malignant_outlier[col_name] \
          > (Q1 - 1.5 * IQR)) & (df_train_malignant_outlier[col_name] < (Q3 + 1.5 * IQR))]
    print(col_name, Q1, Q3, df_train_malignant_outlier.shape)
```

```
('radius_mean', 14.95, 19.55, (146, 32))
('texture_mean', 18.9025, 23.2075, (142, 32))
('perimeter_mean', 97.29500000000002, 128.75, (142, 32))
('area_mean', 679.7, 1172.75, (141, 32))
('smoothness_mean', 0.0926, 0.1099, (138, 32))
('compactness_mean', 0.10679999999999999, 0.166, (136, 32))
('concavity_mean', 0.1056, 0.187, (134, 32))
('concave_points_mean', 0.0610225, 0.0969325, (134, 32))
('symmetry_mean', 0.174225, 0.207875, (132, 32))
('fractal_dimension_mean', 0.0564775, 0.065585, (131, 32))
('radius_se', 0.33925, 0.7201500000000001, (129, 32))
('texture_se', 0.9173, 1.3769999999999998, (122, 32))
('perimeter_se', 2.5545, 4.657, (120, 32))
('area_se', 33.092499999999994, 84.5925, (118, 32))
('smoothness_se', 0.0050392499999999995, 0.007962750000000001, (117, 32))
('compactness_se', 0.01877, 0.03414, (112, 32))
('concavity_se', 0.026465000000000002, 0.044505, (110, 32))
('concave_points_se', 0.010955, 0.015752500000000003, (108, 32))
('symmetry_se', 0.014425, 0.02045, (104, 32))
('fractal_dimension_se', 0.00257325, 0.0042207500000000005, (103, 32))
('radius_worst', 17.6, 23.295, (103, 32))
('texture_worst', 25.785, 31.69, (102, 32))
('perimeter_worst', 116.87499999999999, 152.775, (102, 32))
('area_worst', 935.575, 1654.75, (101, 32))
('smoothness_worst', 0.1281, 0.1563, (101, 32))
('compactness_worst', 0.2376, 0.4116, (101, 32))
('concavity_worst', 0.3158, 0.5006, (101, 32))
('concave_points_worst', 0.1466, 0.198, (101, 32))
('symmetry_worst', 0.2812, 0.3589, (98, 32))
('fractal_dimension_worst', 0.07750249999999999, 0.0970775, (97, 32))
```