Wisconsin Breast Cancer Supervised Learning Project

Data Background

There are 2 categorical variables:

1. ID - specific ID to the patient
2. Diagnosis - “M” for malignant or “B” for benign

Our classification and models will be on the Diagnosis column

There are three classifications of ten features: Mean, Standard Error, and Worst. Mean and standard error are pretty self explanatory, but worst just means the average of the three largest values for each given feature.

There are 30 features total (mean, standard deviation, and worst) of the following 10 features:

1. Radius - calculated by taking the average of the distance of the center of the tumor to the perimeter.
2. Texture - Taking the standard deviation of the grey scale values of the image of the tumor
3. Perimeter
4. Area
5. smoothness - calculated by the variation in radius lengths
6. compactness - calculated as (perimeter^2/area) - 1
7. concavity - severity of the concave portions on the contour image of the tumor
8. concave points - number of concave portions
9. Symmetry
10. fractal dimension ("coastline approximation" - 1)

Features I used:

For the purpose of this project, I decided to limit my features to only 8 out of the 30. I firstly eliminate all the standard error and worst features, only using the mean 10 features. Additionally since the radius, Perimeter and area are all correlated values, I only really have to use one of them. I chose radius for all the tests. This leaves me with the following 8 features for all classification:

1. Radius_mean
2. Texture_mean
3. Smoothness_mean
4. compactness_mean
5. Concavity_mean
6. concave points_mean
7. Symmetry_mean
8. Fractal_dimension_mean

Sample of each class distribution:

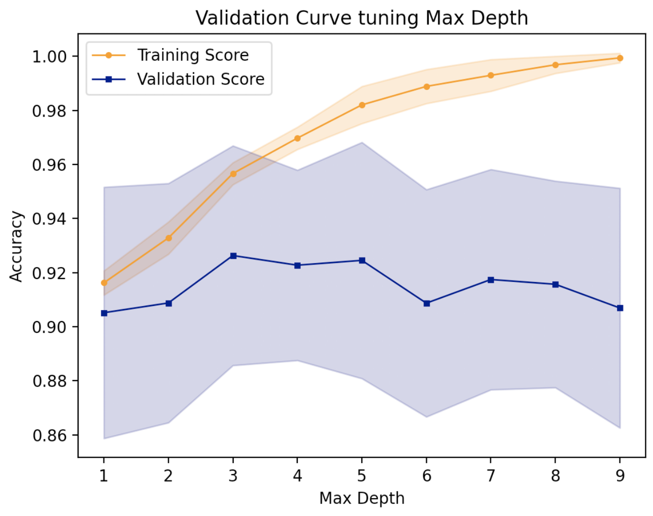
357 benign, 212 malignant for a total of 569 instances.

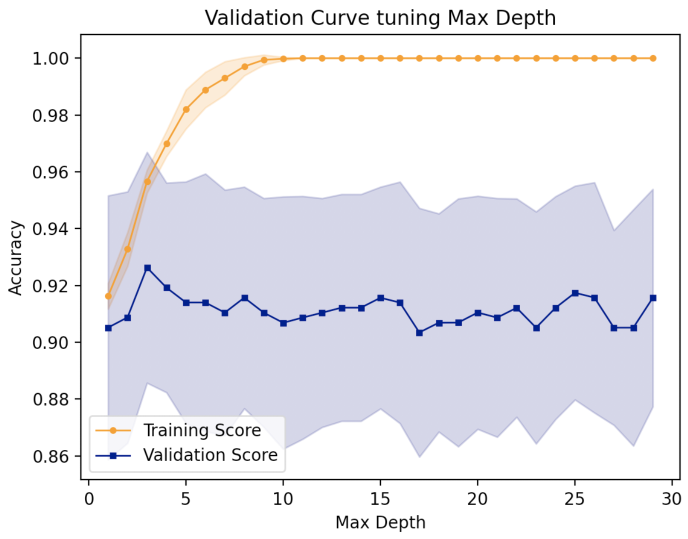
General Approach:

1. Starting set of features: Radius_mean, Texture_mean, Smoothness_mean, compactness_mean, Concavity_mean, concave points_mean, Symmetry_mean, Fractal_dimension_mean (column indices are 2,3,6,7,8,9,10,11).
2. Using these features, tune the hyperparameters for each model. Use validation curves and Gridsearch.
3. Using those hyperparameters returned by GridSearch and the validation curve, use confusion matrices to find the best features of the 8
4. Run a 10-fold cross validation to find the most accurate score for those features and hyperparameters.

Decision Trees:

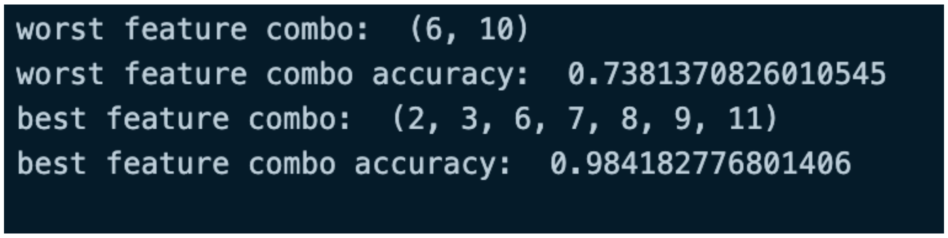
When trying to determine the best features to use for the decision tree classifier, I found that we need to first determine the max depth of the tree. Of course, if we run the Decision tree with max_depth = None (default), all the confusion matrices would show that the classifier got everything correct since it would be overfit to the data. For that reason, I started the decision tree classification by doing the validation curve first. The validation curve shown above demonstrates that despite increasing the depth of the tree, the validation score seems to be generally stagnant (I reran the test with max depth ranging from 1-30). The same result persisted:





After doing some research on the relationship between training and validation scores and how to interpret them, it seems that this is because the data overfits, which makes sense because as the max depth increases we should expect that the training score would approach 1.

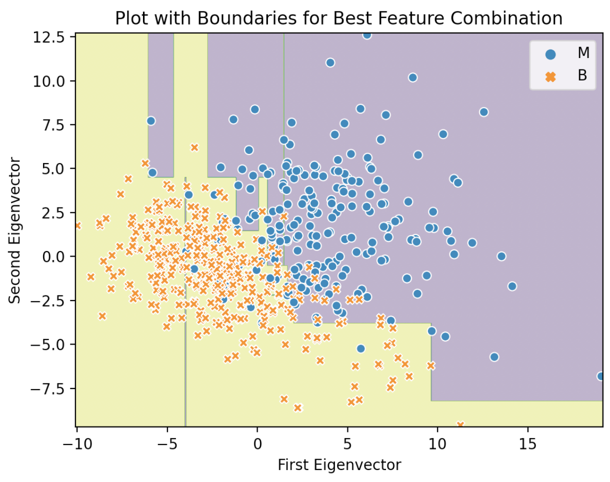
I then used GridSearch, tuning the criterion and max depth features. **The result was that the best criterion was entropy, and max depth 5**. Therefore, I went forward to determine the best and worst features using these hyperparameters in a decision tree. These are the results I got:

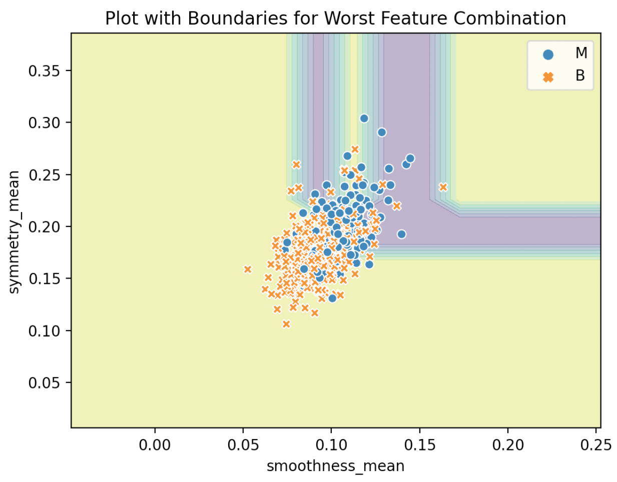


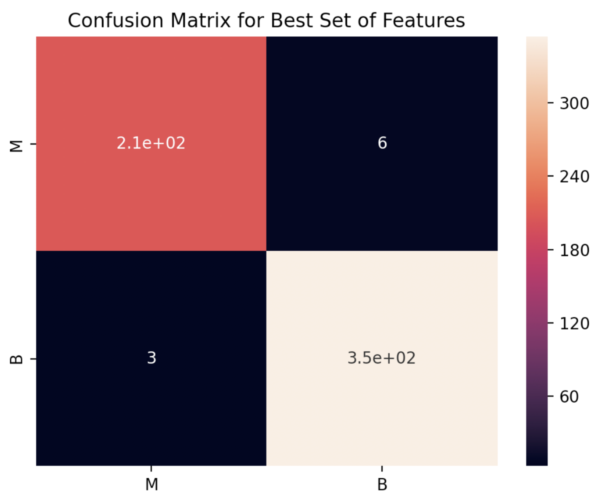
```
worst feature combo:  (6, 10)
worst feature combo accuracy:  0.7381370826010545
best feature combo:  (2, 3, 6, 7, 8, 9, 11)
best feature combo accuracy:  0.984182776801406
```

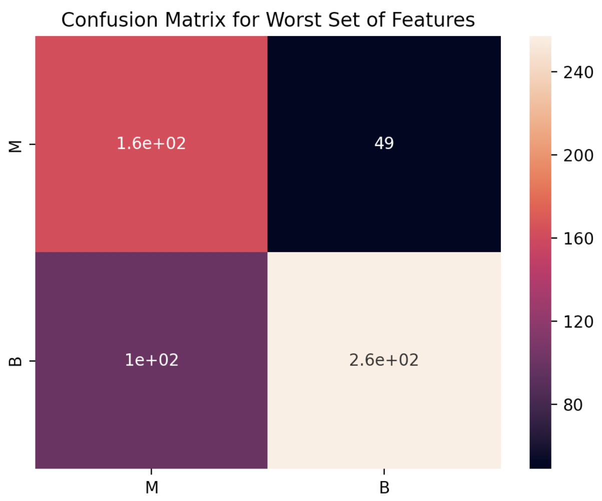
Best combination: Radius_mean, texture_mean, smoothness_mean, compactness_mean, concavity_mean, concave_points_mean, fractal_dimension mean

Worst combination: smoothness_mean, symmetry_mean



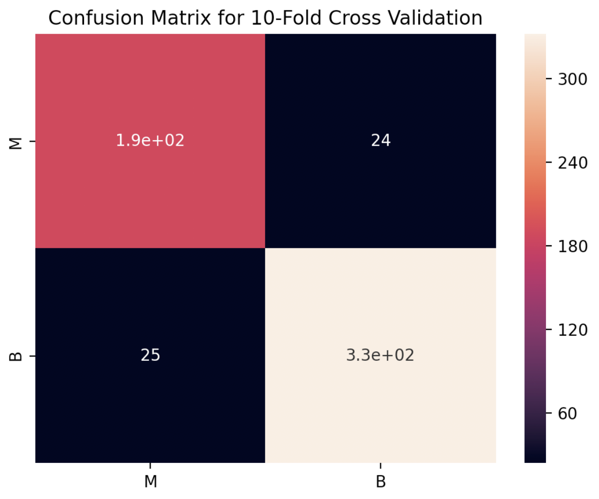






These charts above show the decision boundary plots and confusion matrices for the best and worst combinations of features.

Cross validation:



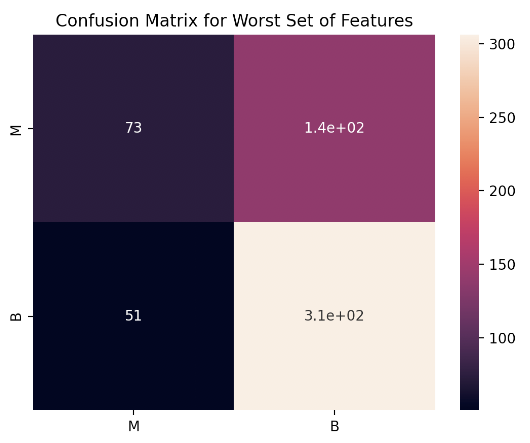
The stratified k-fold accuracy for this model is 0.914 and the training accuracy is 0.972.

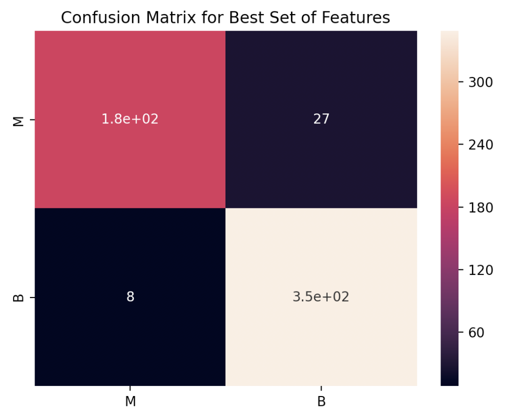
Final Features: Radius_mean, texture_mean, smoothness_mean, compactness_mean, concavity_mean, concave_points_mean, fractal_dimension mean

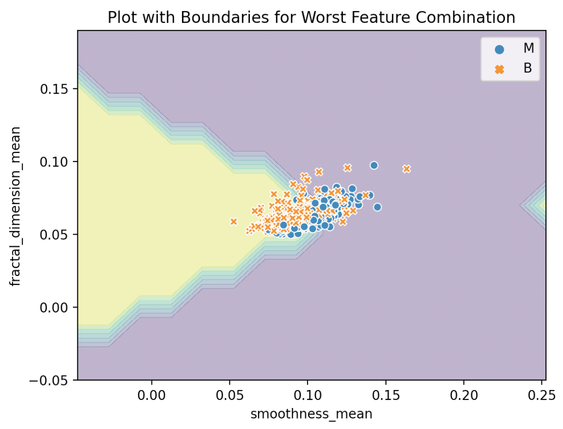
Final Hyper-parameters: criterion is entropy, and max tree depth of 5

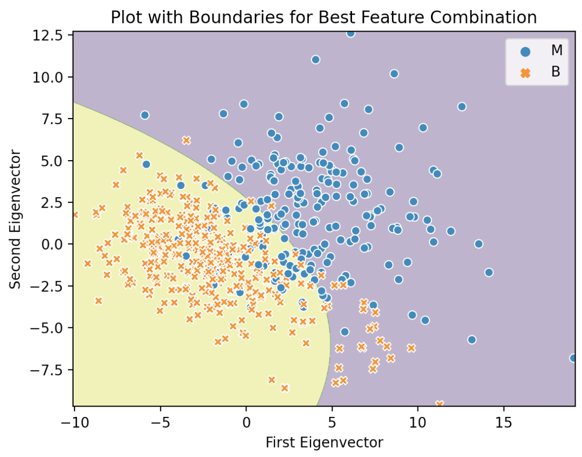
Naive Bayes

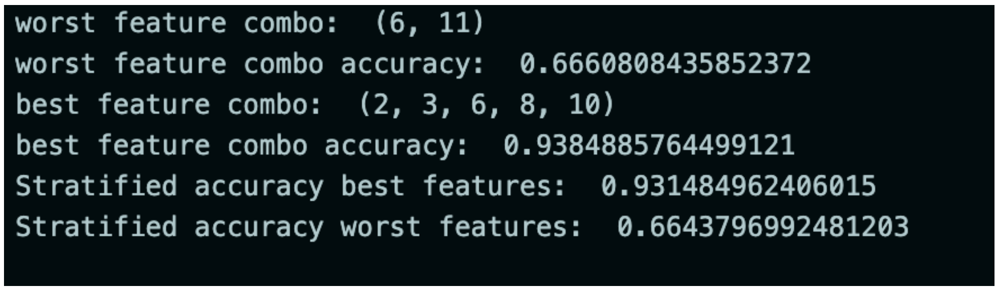
There were no hyper parameters to tune, so here are just the combinations of best and worst features. For the cross validation, I used cv = 10 yielding about 57 items per bucket.







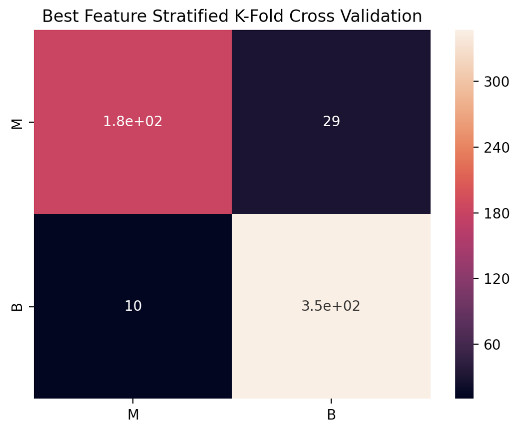




```
worst feature combo:  (6, 11)
worst feature combo accuracy:  0.6660808435852372
best feature combo:  (2, 3, 6, 8, 10)
best feature combo accuracy:  0.9384885764499121
Stratified accuracy best features:  0.931484962406015
Stratified accuracy worst features:  0.6643796992481203
```

<u>Best feature combination:</u> radius_mean, texture_mean, smoothness_mean, concavity mean, symmetry_mean

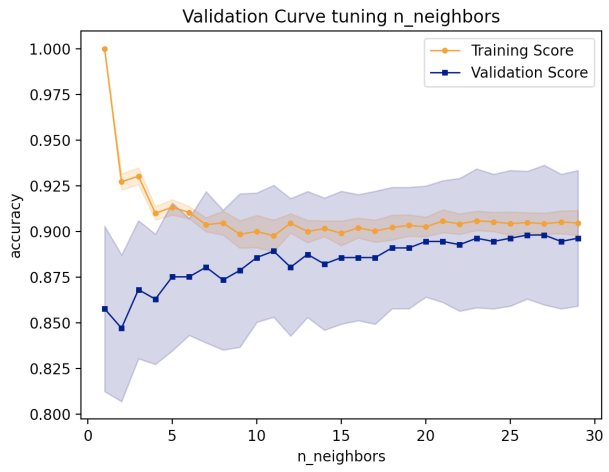
<u>Worst Feature combination:</u> smoothness_mean, fractal_dimension_mean



At left is the k-fold cross validation with an accuracy of 0.931 and a training accuracy of 0.937. **Final model features are:** radius_mean, texture_mean, smoothness_mean, concavity mean, symmetry_mean.

KNN:

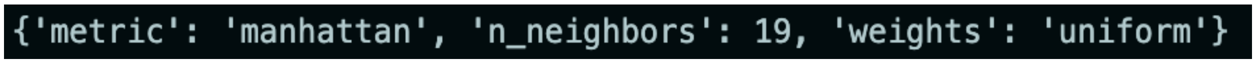
When trying to determine the best KNN model, I started by finding the right hyper parameters. I started first with a validation curve for the n_neighbors hyperparameter:



Interestingly enough, I had done some reading online and a common n_neighbors value is the square root of the total number of instances; in this data the square root would yield about 24, so 19 is within an order of magnitude for that. I also graphed the validation curve shown below:

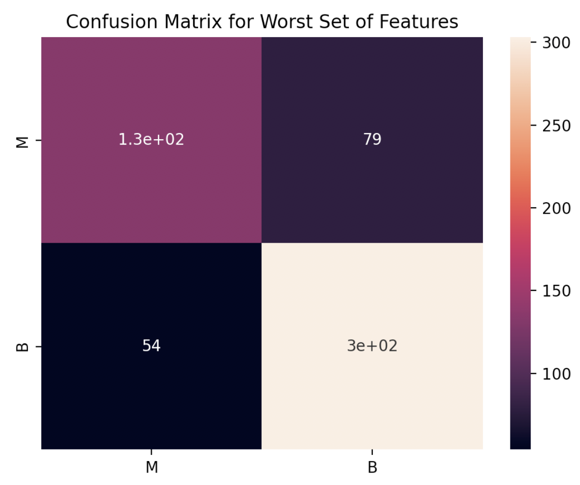
I notice (and of course this makes perfect sense), that initially when bucket size is 0 the training score will be 1. Of course when you're looking at 0 neighbors the classification will make sense.

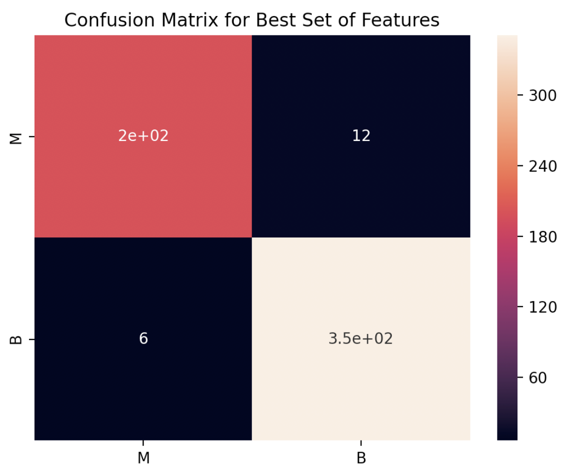
After running grid search on all 8 features, the grid search returned these for the hyper parameters when testing in a range of 1-30 neighbors:

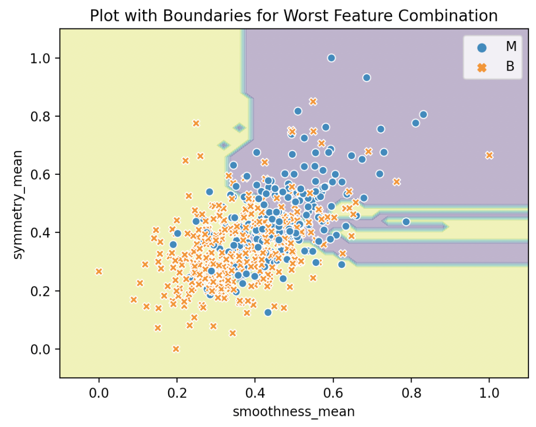


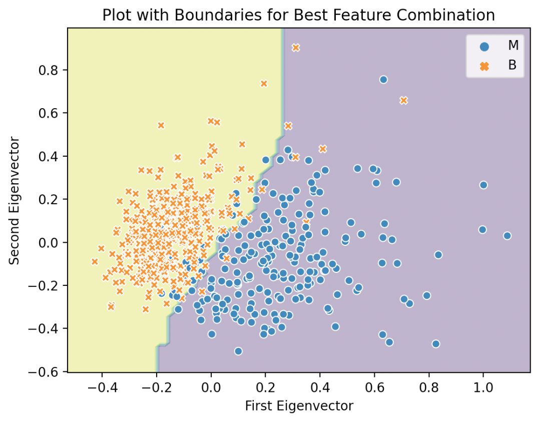
```
{'metric': 'manhattan', 'n_neighbors': 19, 'weights': 'uniform'}
```

Using this information, I then set out to find the best features with the suggested hyperparameters from Gridsearch. I found both the worst and best features:









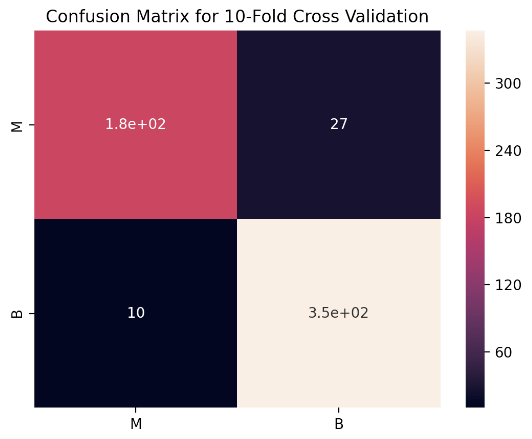
<u>Worst set of features:</u> concavity mean, symmetry_mean

<u>Best set of features:</u> radius_mean, texture_mean, smoothness mean, concavity mean, symmetry mean, fractal_dimension_mean

Final Model:

Features: radius_mean, texture mean, smoothness_mean, concavity_mean, symmetry_mean, and fractal_dimension_mean

Hyper-parameters: manhattan metric, 19 neighbors, uniform weights):



After performing cross validation in the best model (with tuned parameters and features) I get the confusion matrix at left, and the mean cross validation accuracy of this model is 0.935. The mean training accuracy of the model is 0.940.

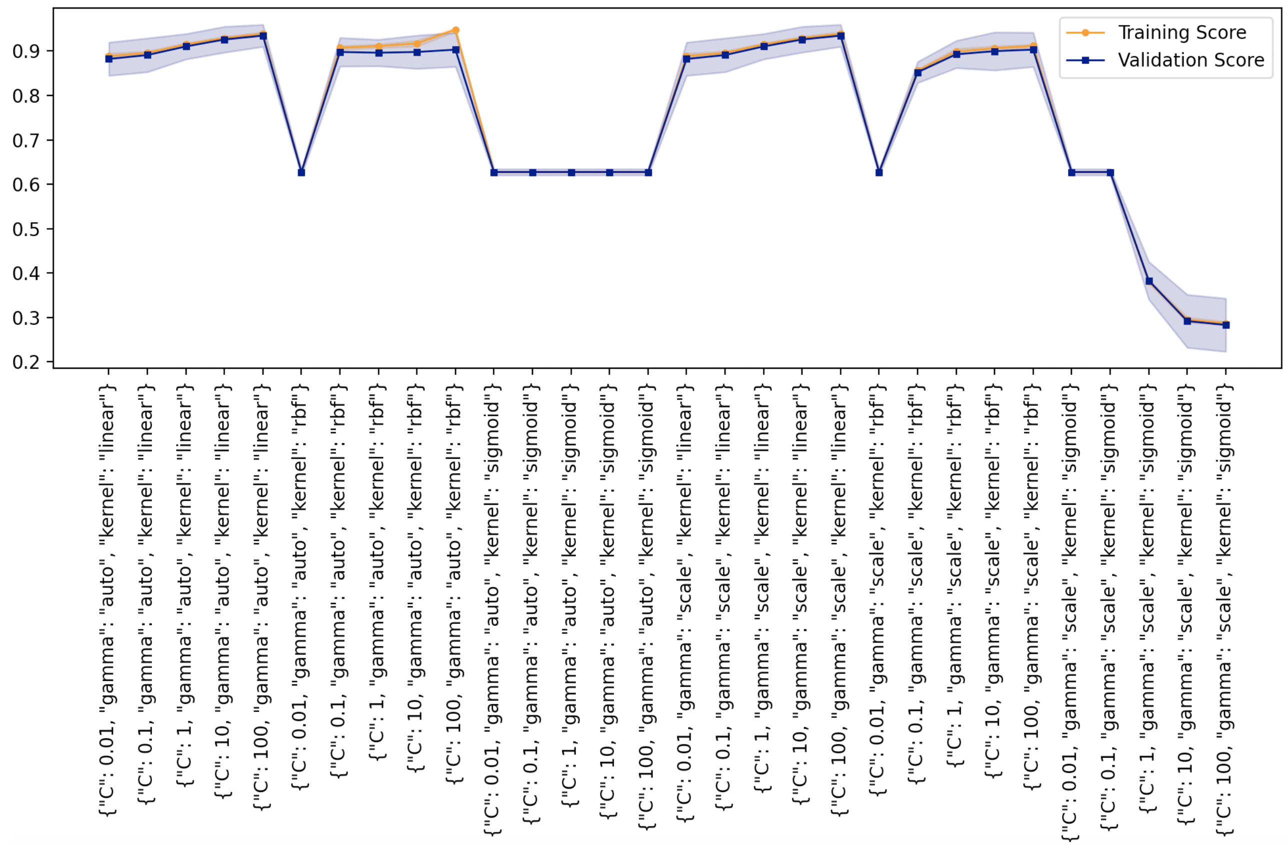
SVM:

I noticed when I first tried to run my Gridsearch with all of the possible kernels, it was incredibly slow to run (I even left it sitting after I went to bed and in the morning it was still not finished running!) Then, I tried excluding one kernel at a time, and noticed that it was the poly kernel that took too long to run. I investigated why and came across a couple suggestions for how to improve the runtime:

- Reduce the dimensionality of your data using PCA
- Reduce the number of samples by using Panda's .sample (n =)

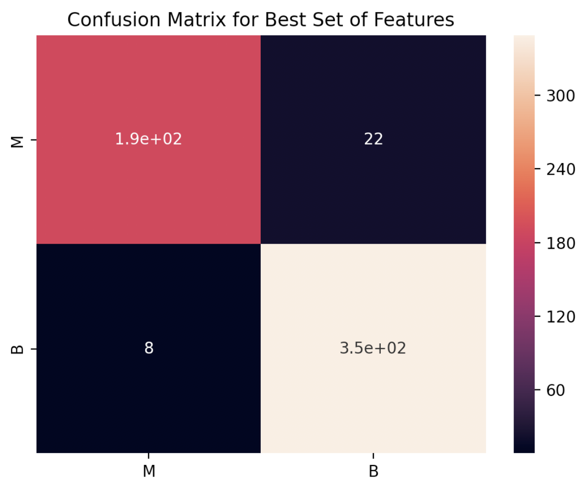
None of these suggestions ended up working for me, but at least I tried. Another note is those suggestions were usually for people who had >10k samples and more than 100 features.

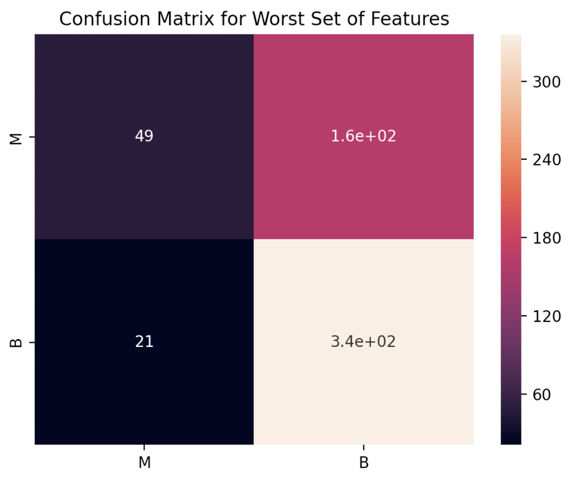
Because we exclude the poly kernel, the degree parameter is not applicable, so I did not do a validation curve using that. However, here is a chart of the training and validation scores for various combinations of parameters.



Using GridSearch, I found the best combination of parameters was C: 100, Gamma: scale, kernel: linear

I then used this combination of hyperparameters to find the best set of features:





<ins>**Best set of features:**</ins> radius_mean, texture_mean, smoothness_mean, compactness_mean, concavity_mean, concave points_mean

Accuracy: 0.947 (not cross validated)

<ins>**Worst set of features:**</ins> smoothness_mean, fractal_dimension_mean

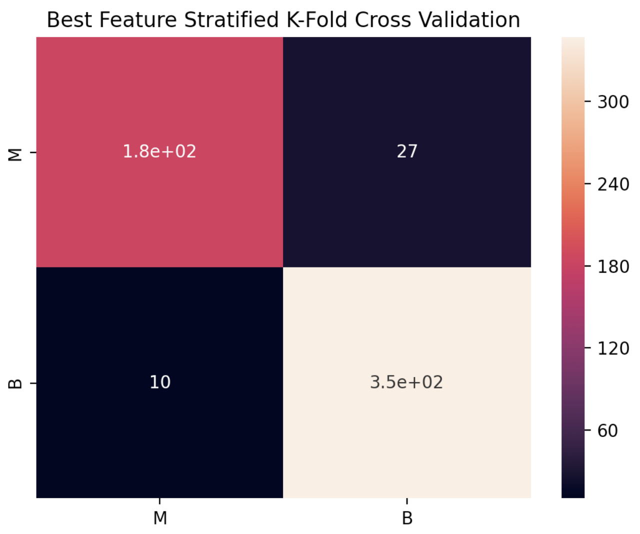
Accuracy: 0.677 (not cross validated)

Final model:

Features: radius_mean, texture_mean, smoothness_mean, compactness_mean, concavity_mean, concave points_mean

Hyper parameters: C: 100, Gamma: scale, kernel: linear

Stratified K Fold:



I ran stratified k-fold using the best SVM hyperparameter combination with the best feature combination (radius_mean, texture_mean, smoothness_mean, compactness_mean, concavity_mean, concave points_mean) to get the confusion matrix above.

The stratified k-fold accuracy for this model is 0.936. The training score is 0.943.

Summary Table of 4 Final Tuned models:

	K-Fold Accuracy	**Training Accuracy**	**Hyper-parameters**	**Features**
Decision Tree	0.914	0.972	criterion: entropy max depth: 5	Radius_mean, texture_mean, smoothness_mean, compactness_mean, concavity_mean, concave_points_mean, fractal_dimension mean
KNN	0.935	0.94	Metric: manhattan n_neighbors: 19 weights: uniform	radius_mean, texture_mean, smoothness_mean, concavity_mean, symmetry_mean, and fractal_dimension_mean
Naive Bayes	0.931	0.937	N/A	radius_mean, texture_mean, smoothness_mean, concavity mean, symmetry_mean

SVM	0.936	0.943	C: 100 Gamma: scale kernel: linear	radius_mean, texture_mean, smoothness_mean, compactness_mean, concavity_mean, concave points_mean

Interesting Findings Summary:

I made this section mostly for my own reference on interesting/helpful things I found in my online searches throughout the duration of the project.

- An increasing training score and stagnant validation score indicates an overfit of data.
    - Some notes about interpreting training scores vs validation scores:
        - A super high training score in data with high variance (overfitting) may result in a low test score.
        - A close training accuracy and test score is an indicator for a good model.
- From the SVM parameter tuning, if certain kernels are taking a long time try these:
    - Reduce the dimensionality of your data using PCA
    - Reduce the number of samples by using Panda's sample function.
- A common starting point for guessing n_neighbors parameter for KNN is sqrt(instances). In this case, there are 569 instances total, giving us a guess of about 24 neighbors.
    -

Conclusion:

From these 4 models, the one with the highest cross validation score was the tuend SVM model, with a score of 0.936, and a training score of 0.943. This is good since the test score and training score are relatively close, often an indicator of a good model. To be honest, all 4 of these models are quite similar in terms of their scores (with maybe the decision tree as an exception). If I had more time, I could get better results by running a gridsearch on every combination of features for each of the 4 models to determine the best feature + hyperparameter feature. My approach of using the same 8 features to find the hyperparameters is a good estimate of which hyperparameters would be best, but probably not as accurate as I could have gotten running GridSearch on everything.

An interesting observation I made was that I noticed the worst feature combination in most of the models was smoothness_mean in combination with either symmetry_mean or fractal_dimension_mean. I suspect this could be because by themselves without a sizing feature (such as radius), they don’t mean much on their own (I’m not a biology expert but I’d imagine tumor growth and size would have something to do with whether it is malignant or benign to a certain extent). This is also probably why every single best feature combination included radius.

I have neither given nor received any unauthorized aid on this assignment.

[illegible]