**Introduction.**

**This data set we are exploring the Wisconsin Breast Cancer Data Set with the use of different machine learning models and methods. We picked this data set for our project due to the wide impact breast cancer has on society; just in the UK, breast cancer accounts for as approximately 15% of all new cancer cases, resulting in over 150 new diagnosis a day.**

**Our project we explore at the issue of differentiating between malignant (cancerous) and benign (noncancerous) tumours. A good indicator of a tumour being malignant is its shape, however, there is a lot of variation with every tumour which can make it difficult to differentiate between malignant and benign tumours accurately. Detecting malignant tumours early would benefit patient treatment, mortality rate and reduce costs. With machine learning, we can identify and predict results based on patterns that humans would not be able to identify. As due to the nature of cancer the cost of false negatives is significantly higher, so we will consider this when training our models by trying to reduce our false negatives.**

**Our project will try to determine, using machine learning models, if a tumour is malignant or benign based on variables corresponding to its shape. The data set has 562 instances, with the label M or B for the type of cancer and 10 unique features (each with 2 more features containing its SD and Worst (largest) value. These include…**

* Radius
* Texture
* Perimeter
* Area
* Smoothness
* Compactness
* Concavity
* Concave Points
* Symmetry
* Fractal Dimension

**Data Pre-processing**

1. Observe dataset, add header
2. Look at data types in
   1. Dropped ID column, as not bearing on our models
   2. LabelEncoder on label, from object -> int (categorical) M = 1 B = 0
   3. Tested for any null values, none found
3. Explore our data using plots
   1. Count & Pie plots of B/M
   2. Summary statistics
   3. Pair plot -> Talk about leading diagonal
   4. Violin Plot (standardised) talk about distribution
   5. Correlation graph -> Concavity concave, area, perimeter, radius e.c.t point very correlated
4. Drop Perimeter and Area features, as they are very heavily (mathematical) correlated and are function of the radius. Also, most breast cancer is a circular shape(ish), so this supports our method. Maybe drop some of concavity, concave points and compactness e.c.t?
5. Standardise data, as some algorithms require it and the one that don’t, don’t care about it.
6. Use SelectKBest to reduce our instances, as we only have ~500 instance and ~30 features- this will help us reduce overfitting our data. How many variables do we want?
   1. We aren’t using PCA, as we want to be able to make inferences on our data.
7. Split our data into the train/test /validation sets. Stratify split as this is a classification problem

# Put in some graphs, not many and only a few using pair plot -maybe skip out on violin plot.

**Forest Algorithm**

* Create initial model and train using train and validation set. Use it to get feature importance, consider we do not want many false negatives, so we want a high recall value. We consider overfitting as we have such a high accuracy?
* Remove least important features, <~1%?
* Repeat benchmarks on new features.
* Using random grid and randomised CV, we repeat this code with differing parameters until our model sticks to one ‘vibe’ of values. Explain how the tuning process works with different trees e.c.t and explain some of the parameters and how they issue our model.
* Now we have good metrics, and we want to test for over/underfitting. Use learning curve to explain how we think it is/ isn’t, also talk about how the graph changes when we reduce the number of variables significantly.

!!!! should keep sample instances the same when I re split train valid test sets!!!!!!!

Probably equivalent to one page, so we can use Roc for model comparison.

**Model Comparison & Discussion**

* Compare the final metrics (accuracy, recall, precession) of our models
* Compare using the Roc, talk about false positive in results
* How we would want more instances, overfitting.
* More domain knowledge about how the different features work with each other, instead of just depending on the correlation graph.

**K NEAREST NEIGHBOURS**

* Once data is processed, fit the model
* Pick 2 features to focus on since KNN works best with fewer features (I remember someone saying, will factcheck)
* Explain how KNN works and how it fits the data were looking at
* Plot accuracy against the hyperparameter k and pick the k s.t we have the model with the highest accuracy
* Once we have got this k train the model again but with this k and use the graph along with other information to see visually whether the model is overfitting or underfitting our data
* Use in conjunction with learning curves
* Little idea, plot with multiple features to see at what point having too many affects the accuracy
* Followed up by model comparison

!!!!NEED TO SET RANDOM STATE BEFORE FIGURES USED!!!!!