**Self-Review**

**Questions:**

* What?
* Why?
* How?
* When?

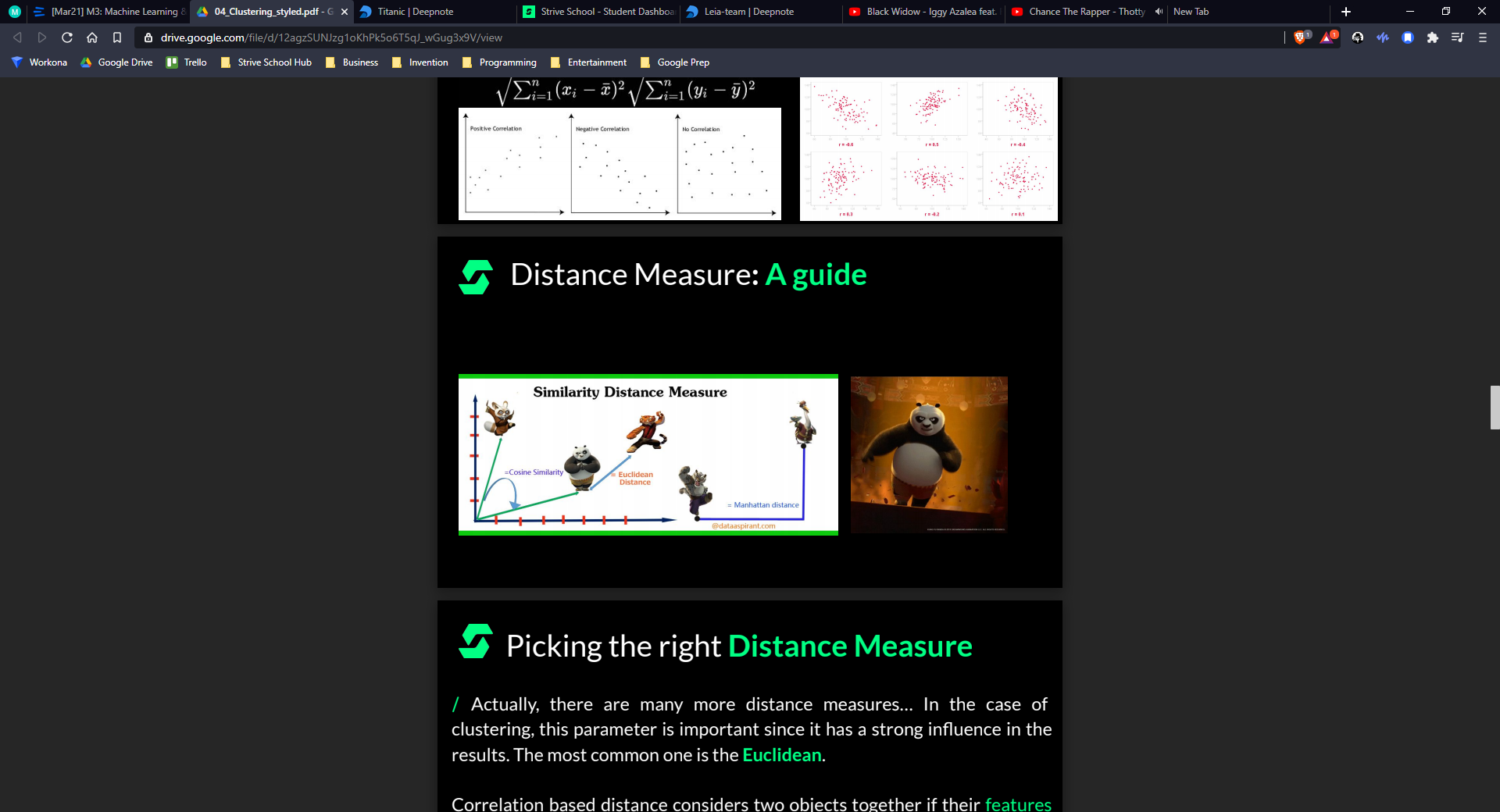
**Key Topics:**

* ~~KMeans~~
* k-NN
* Train Test Split
  + x\_train, x\_test, y\_train, y\_test
  + X, y
* ~~PCA~~
* ~~Clustering~~
* Linear and logistic regression
* Decision trees
* SVM
* OneHotEncoding

**PCA:**

1. What is PCA?
   1. PCA stands for **P**rincipal **C**omponent **A**nalysis. It finds the most equally distanced line between all data points [DOES IT CONSIDER OUTLIERS? – PCA is sensitive to outliers, you must deal with them before applying PCA].
   2. Used to reduce the dimensions of a dataset, making other ML algorithms take less computational power to process the data [IS THIS WHY WE DO IT? IS THIS TRUE?]. Reducing dimensions comes at the cost of losing portions of data.
      1. For every dimension reduced you lose portions of data
   3. PCA must standardize the data [WHAT DOES STANDARDIZE MEAN? – Standardize means to make the data conform to the same scale. If you have one set of data that goes up to 100 and another column that goes to 10, AI will put more weight on the 100 column.] before it can work with it. If it doesn’t it will put more value on data with higher values, purely because the number is bigger. ~~something with 100 attached to it than it would to a data point of value 1.~~
   4. There needs to be a strong correlation for PCA to effectively perform dimension (and data) reduction. Using a correlation heatmap helps to see if PCA will work, if the correlation is less than 0.3, PCA will not work well.
      1. Negatively correlated points will also work (-0.3 or more)
      2. **KEY is PCA needs a strong correlation to perform dimension reduction.**
2. Why use PCA?
   1. Datasets with many dimensions and a lot of data also hold a lot of unnecessary information. **Added dimensions adds empty space**.
   2. Reducing dimensions on data with strong correlation = Easier to deal with data.
      1. Less dead space
      2. Easier to plot (Hard to plot a 26-dimension dataset, but a 3-dimensional dataset can be visualized)
      3. Less dimensions means less processing time for ML algorithms.
3. How to use PCA?
   1. **from** **sklearn.decomposition** **import** PCA
   2. X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
   3. pca = PCA(n\_components=2)
      1. n\_componenets is how many dimensions you want to reduce?
   4. pca.fit(X)
      1. .fit means training. You are taking the ML model and training it with .fit
      2. .fit turns the data into weights, this allows for more accuracy in the ML model.
4. When do you use PCA?
   1. **When you have a large multidimensional dataset with high correlation**.

**Clusters:**

1. LDA vs PCA:
   1. LDA needs labels, if dealing with a huge dataset you cannot perform LDA.
   2. PCA does not need labels but loses more data?
   3. **KEY : If possible, use LDA**.
2. Process:
   1. **Raw data** -> Clustering algorithm -> **Clusters of data**
3. Types of clustering:
   1. Hierarchical, hight of y value
   2. Centre of data clusters, use KMeans for this.
4. Measuring distance between clusters:
   1. Main method = Euclidean distance. Straight line from one data point to another
   2. Manhattan method – Goes up or down, left or right. Like driving around New York’s block system
   3. Photo to show methods:
      1. 

**KMeans:**

1. What is KMeans?
   1. Calculates the centre of clusters to distinguish clusters of data.
   2. You must specify how many clusters you want.
      1. If cluster amount selected is the same as number of data points, then you get a Voronoi diagram.
   3. Unsupervised model
   4. **KMeans makes no assumptions, just specify the number of clusters, and let it work on the data**.
2. Why use KMeans?
   1. An effective way to take messy data and organise into clear clusters.
      1. If you can see there are 3 clusters, then tell KMeans to split into 3 and it will do the rest.
3. How to use KMeans?
   1. **from** **sklearn.cluster** **import** KMeans
   2. **import** **numpy** **as** **np**
   3. X = np.array([[1, 2], [1, 4], [1, 0],
   4. [10, 2], [10, 4], [10, 0]])
   5. kmeans = KMeans(n\_clusters=2, random\_state=0).fit(X)
      1. n\_clusters is how many clusters you want
      2. random\_state must be set to the same number everytime. You can pick any number
      3. .fit trains the model
   6. kmeans.labels\_
      1. Prints the labels (NOT NESSASARY)
   7. kmeans.predict([[0, 0], [12, 3]])
      1. .predict will try to predict the cluster the given array would fall in.
   8. kmeans.cluster\_centers\_
      1. Prints cluster centres (NOT NESSASARY)
4. When to use KMeans?
   1. When there are clear clusters visually, and you want to split them into their defined numbers.
   2. WHAT IF YOU CAN’T SEE CLEAR CLUSTERS? WILL KMEANS MESS IT UP? – Yes, it just splits them up into what it perceives as clusters. If the clusters are not clear, then you will get strange splits.
   3. [Placeholder]

**Train Test Split:**

1. What is train\_test\_split?
   1. Split into x train, x test, y train, y test.