Homework 1

Week 1 - Clustering and Regression

Patthadon Phengpinij

Collaborators. ChatGPT

1 Metrics

In a population where the amount of cats is equal to the amount of dogs. Considering the following classification results from a classifier.

Model A	Predictied dog	Predictied cat
Actual dog	30	20
Actual cat	10	40

T1. What is the accuracy of Model A?

Solution. First, assume that we consider dogs as 'class 0' (negative) and cats as 'class 1' (positive). From the confusion matrix above, we can identify the following values:

- True Positives (TP): 40 (Actual cat predicted as cat)
- True Negatives (TN): 30 (Actual dog predicted as dog)
- False Positives (FP): 10 (Actual cat predicted as dog)
- False Negatives (FN): 20 (Actual dog predicted as cat)

The formula for accuracy is given by:

$$\begin{aligned} Accuracy &= \frac{TP + TN}{TP + TN + FP + FN} \\ &= \frac{40 + 30}{40 + 30 + 10 + 20} \\ &= \frac{70}{100} \\ Accuracy &= 0.7 \end{aligned}$$

Thus, from the calculation above, the **Accuracy** of Model A is $\boxed{\text{0.7 or 70\%}}$.

T2. Consider cats as 'class 1' (positive) and dogs as 'class 0' (negative), calculate the precision, recall, and F1.

Solution. The situation is the same as in Problem T01, where we have all TP, TN, FP, and FN values identified. Now, we can calculate precision, recall, and F1 score using the following formulas:

$$\begin{aligned} Precision &= \frac{TP}{TP + FP} \\ Recall &= \frac{TP}{TP + FN} \\ F1 &= \frac{2TP}{2TP + FP + FN} \end{aligned}$$

Substituting the values we have:

$$\begin{aligned} Precision &= \frac{40}{40 + 10} \\ &= \frac{40}{50} \\ Precision &= 0.8 \\ Recall &= \frac{40}{40 + 20} \\ &= \frac{40}{60} \\ Recall &\approx 0.6667 \\ F1 &= \frac{(2 \times 40)}{(2 \times 40) + 10 + 20} \\ &= \frac{80}{80 + 30} \\ &= \frac{80}{110} \\ F1 &\approx 0.7273 \end{aligned}$$

Thus, the calculated metrics are:

- **Precision:** $\boxed{0.8}$
- **Recall:** $\boxed{\approx 0.6667}$
- **F1 Score:** $\boxed{\approx 0.7273}$

T3. Consider class cat as 'class 0' and class dog as 'class 1', calculate the precision, recall, and F1.

Solution. Switching the classes means we need to redefine TP, TN, FP, and FN:

- True Positives (TP): 30 (Actual dog predicted as dog)
- True Negatives (TN): 40 (Actual cat predicted as cat)
- False Positives (FP): 20 (Actual dog predicted as cat)
- False Negatives (FN): 10 (Actual cat predicted as dog)

Now, we can calculate precision, recall, and F1 score using the same formulas:

$$\begin{aligned}
Precision &= \frac{TP}{TP+FP} \\
Recall &= \frac{TP}{TP+FN} \\
F1 &= \frac{2TP}{2TP+FP+FN}
\end{aligned}$$

Substituting the new values:

$$\begin{aligned}
Precision &= \frac{30}{30+20} \\
&= \frac{30}{50} \\
Precision &= 0.6 \\
Recall &= \frac{30}{30+10} \\
&= \frac{30}{40} \\
Recall &= 0.75 \\
F1 &= \frac{(2\times 30)}{(2\times 30)+20+10} \\
&= \frac{60}{60+30} \\
&= \frac{60}{90} \\
F1 &\approx 0.6667
\end{aligned}$$

Thus, the calculated metrics are:

- **Precision:** $\boxed{0.6}$
- **Recall:** $\boxed{0.75}$
- **F1 Score:** $\boxed{\approx 0.6667}$

Note: It is important to specify the 'positive' class when you calculate precision, recall, and F1. If there are more than two classes, it is usually done in a one versus-all setting where one class is considered positive and the rest of the classes are considered negative.

T4. Now consider a lopsided population where there are 80% cats. What is the accuracy of Model A? Using dog as the positive class, what is the precision, recall, and F1? Explain how and why these numbers change (or does not change) from the previous questions.

Solution. In a lopsided population where 80% are cats and 20% are dogs, we need to adjust the confusion matrix accordingly. Assuming the same classification performance as Model A, we can scale the confusion matrix based on the new population distribution.

Model A	Predictied dog	Predictied cat
Actual dog	12	8
Actual cat	16	64

Explaining the changes:

- True Positives (TP): 12 (Actual dog predicted as dog)
- True Negatives (TN): 64 (Actual cat predicted as cat)
- False Positives (FP): 16 (Actual cat predicted as dog)
- False Negatives (FN): 8 (Actual dog predicted as cat)

This new confusion matrix came from scaling down the previous counts by a factor of 0.4 for dogs (since the dog population decreased from 50 to 20) and scaling up by a factor of 1.6 for cats (since the cat population increased from 50 to 80).
Next, we calculate the accuracy, precision, recall, and F1 score:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{12 + 64}{12 + 64 + 16 + 8} = \frac{76}{100} = 0.76$$

$$Precision = \frac{TP}{TP + FP} = \frac{12}{12 + 16} = \frac{12}{28} \approx 0.4286$$

$$Recall = \frac{TP}{TP + FN} = \frac{12}{12 + 8} = \frac{12}{20} = 0.6$$

$$F1 = \frac{(2 \times 12)}{(2 \times 12) + 16 + 8} = \frac{24}{24 + 24} = \frac{24}{48} = 0.5$$

Thus, the calculated metrics in the lopsided population are:

- **Accuracy:** $\boxed{\text{0.76 or 76\%}}$
  The accuracy has increased from 0.7 to 0.76 because the model correctly classifies a larger proportion of the majority class (cats).
- **Precision:** $\boxed{\approx 0.4286}$
  Precision has decreased significantly because the number of false positives (cats predicted as dogs) has increased relative to true positives.
- **Recall:** $\boxed{0.6}$
  Recall has decreased slightly from 0.6667 to 0.6, indicating that the model is missing more actual dogs.
- **F1 Score:** $\boxed{0.5}$
  The F1 score has decreased from approximately 0.7273 to 0.5, reflecting the trade-off between precision and recall in this lopsided population.

OT1. Consider the equations for accuracy and F1

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$F1 = \frac{2TP}{2TP + FP + FN}$$

When will accuracy be equal, greater, or less than F1?

Solution. To determine when accuracy is equal to, greater than, or less than F1, we can set up the equations and analyze them. Setting accuracy equal to F1:

$$\begin{aligned} Accuracy &= F1 \\ \frac{TP + TN}{TP + TN + FP + FN} &= \frac{2TP}{2TP + FP + FN} \\ (TP + TN)(2TP + FP + FN) &= (2TP)(TP + TN + FP + FN) \end{aligned}$$

Expanding both sides, continuing the simplification, we get:

$$0 = (TP - TN)(FP + FN)$$

From the final equation, we can see that **accuracy equals F1** when either:

- $TP = TN$ (the number of true positives equals the number of true negatives), or
- $FP + FN = 0$ (since both FP and FN cannot be negative, there are no false positives or false negatives, meaning **perfect classification**).

Next, we analyze when accuracy is greater than F1:

$$\begin{aligned} Accuracy &> F1 \\ \frac{TP + TN}{TP + TN + FP + FN} &> \frac{2TP}{2TP + FP + FN} \end{aligned}$$

Following similar steps as above, we got:

$$0 > (TP - TN)(FP + FN)$$

Since $FP + FN$ is always non-negative, assume imperfect classification, for the product to be negative, we must have:

$$\begin{aligned} TP - TN &< 0 \\ TP &< TN \end{aligned}$$

Thus, we find that **accuracy is greater than F1** when:

$TP < TN$ (the number of true positives is less than the number of true negatives)

On the other hand, following similar steps, we find that **accuracy is less than F1** when:

$TP > TN$ (the number of true positives is greater than the number of true negatives)

Therefore, summarizing the results:

- $Accuracy = F1$ when $TP = TN$ or $FP + FN = 0$ (perfect classification).
- $Accuracy > F1$ when $TP < TN$ and $FP + FN > 0$ (imperfect classification).
- $Accuracy < F1$ when $TP > TN$ and $FP + FN > 0$ (imperfect classification).

2 Clustering

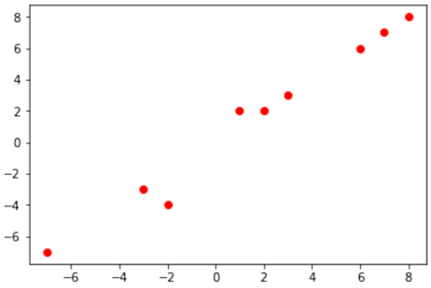
Recall from lecture that K-means has two main steps: the points assignment step, and the mean update step. After the initialization of the centroids, we assign each data point to a centroid. Then, each centroids are updated by re-estimating the means.

Concretely, if we are given N data points, x_1, x_2, ..., x_N, and we would like to form K clusters. We do the following;

1. **Initialization:** Pick K random data points as K centroid locations c_1, c_2, ..., c_K.
2. **Assign:** For each data point k, find the closest centroid. Assign that data point to the centroid. The distance used is typically Euclidean distance.
3. **Update:** For each centroid, calculate the mean from the data points assigned to it.
4. **Repeat:** Repeat step 2 and 3 until the centroids stop changing (convergence).

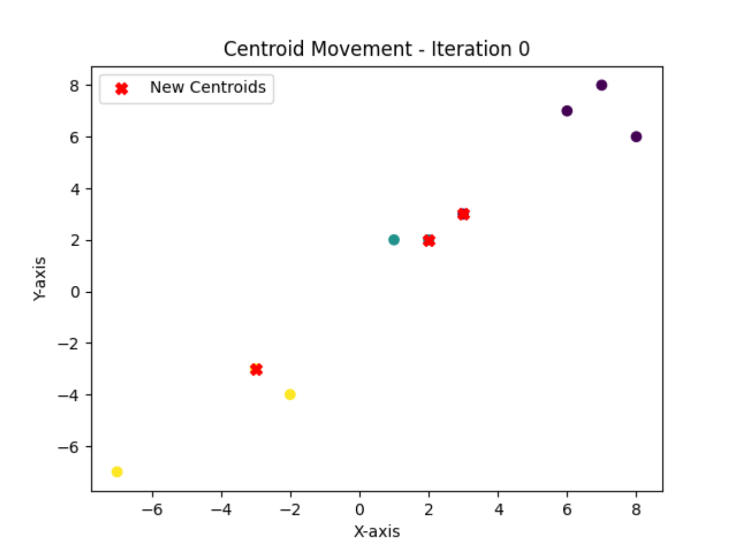
Given the following data points in x-y coordinates (2 dimensional)

x	y
1	2
3	3
2	2
8	6
7	8
6	7
-3	-3
-2	-4
-7	-7



T5. If the starting points are $(3,3)$, $(2,2)$, and $(-3,-3)$. Describe each assign and update step. What are the points assigned? What are the updated centroids? You may do this calculation by hand or write a program to do it.

Solution. First, we start with the initial centroids:



Apply the K-means algorithm step by step with the given starting centroids $(3,3)$, $(2,2)$, and $(-3,-3)$. The steps are as follows:

x	y	Assigned Centroid
1	2	$(2,2)$
3	3	$(3,3)$
2	2	$(2,2)$
8	6	$(3,3)$
7	8	$(3,3)$
6	7	$(3,3)$
-3	-3	$(-3,-3)$
-2	-4	$(-3,-3)$
-7	-7	$(-3,-3)$

Update the centroids based on the assigned points:

- For centroid $(3,3)$:

$$\text{New Centroid} = \left(\frac{3+8+7+6}{4}, \frac{3+6+8+7}{4}\right) = (6,6)$$

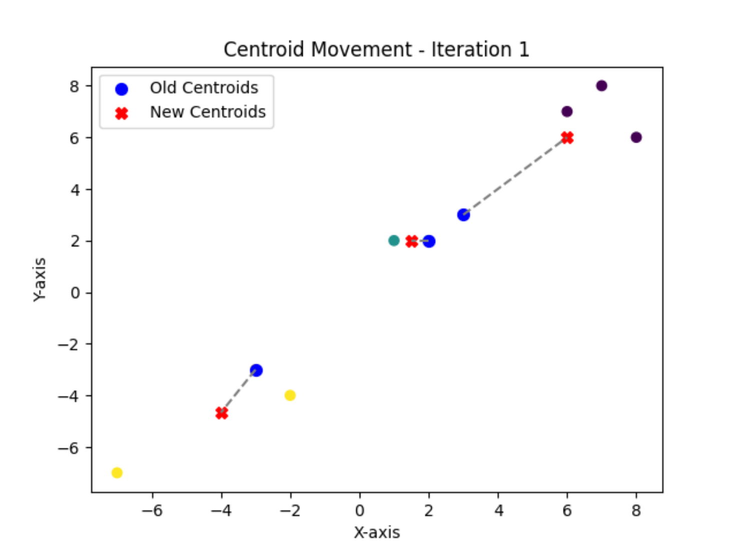
- For centroid $(2,2)$:

$$\text{New Centroid} = \left(\frac{1+2}{2}, \frac{2+2}{2}\right) = (1.5,2)$$

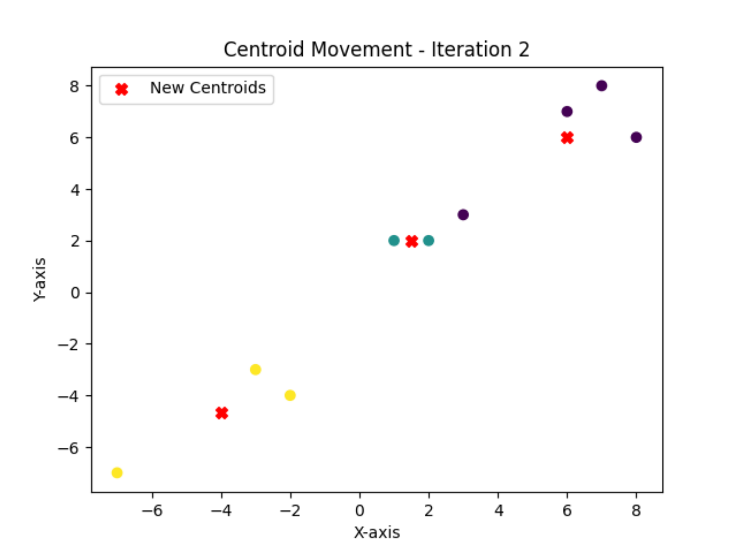
- For centroid $(-3,-3)$:

$$\text{New Centroid} = \left(\frac{-3+-2+-7}{3}, \frac{-3+-4+-7}{3}\right) = (-4,-4.67)$$

After the first iteration, the updated centroids are:



Continuing the K-means algorithm with the new centroids $(6, 6)$, $(1.5, 2)$, and $(-4, -4.67)$, we got:



From the second iteration, the centroids do not change anymore, indicating convergence. Thus, the final centroids after convergence are:

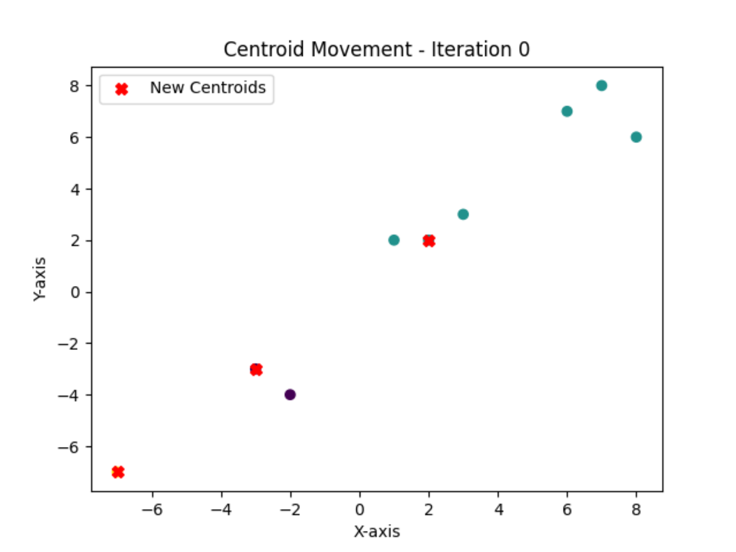
- Centroid 1: $(6, 6)$
- Centroid 2: $(1.5, 2)$
- Centroid 3: $(-4, -4.67)$

The points assigned to each centroid are:

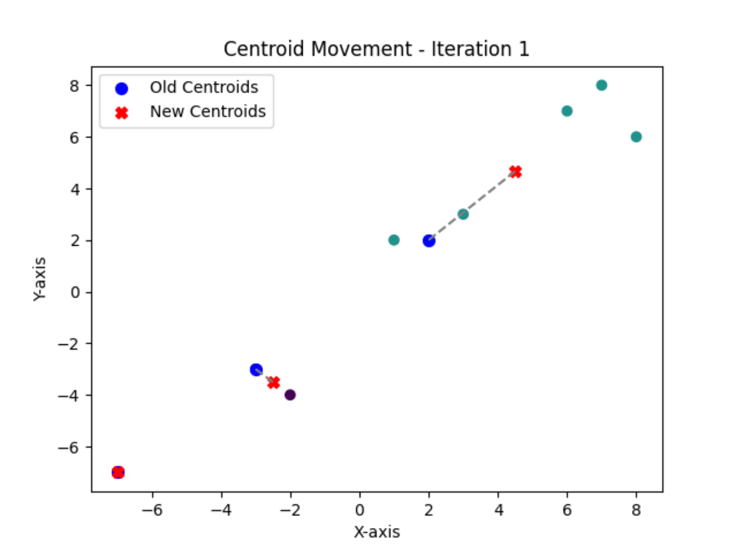
x	y	Assigned Centroid
1	2	$(1.5, 2)$
3	3	$(1.5, 2)$
2	2	$(1.5, 2)$
8	6	$(6, 6)$
7	8	$(6, 6)$
6	7	$(6, 6)$
-3	-3	$(-4, -4.67)$
-2	-4	$(-4, -4.67)$
-7	-7	$(-4, -4.67)$

T6. If the starting points are $(-3, -3)$, $(2, 2)$, and $(-7, -7)$, what happens?

Solution. Just like in Problem T05, we start with the initial centroids:



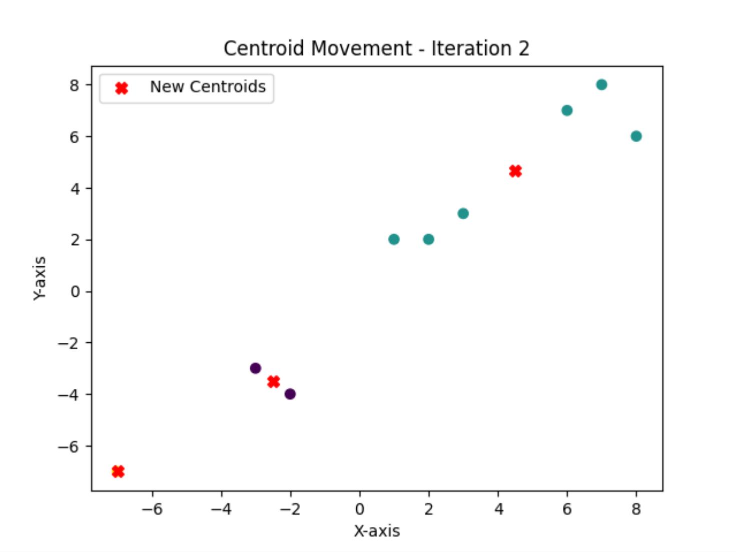
Applying the K-means algorithm step by step with the given starting centroids $(-3, -3)$, $(2, 2)$, and $(-7, -7)$. The first iteration results shown below:



After the first iteration, the updated centroids are:

- For centroid $(-3,-3)$:
$$\text{New Centroid} = \left(\frac{-3+-2}{2}, \frac{-3+-4}{2}\right) = (-2.5, -3.5)$$
- For centroid $(2,2)$:
$$\text{New Centroid} = \left(\frac{1+3+2+8+7+6}{6}, \frac{2+3+2+6+8+7}{6}\right) = (4.5, 4.67)$$
- For centroid $(-7,-7)$:
$$\text{New Centroid} = (-7,-7)$$
(a point itself)

Continuing the K-means algorithm with the new centroids $(-2.5,-3.5)$, $(4.5,4.67)$, and $(-7,-7)$, we got:



From the second iteration, the centroids do not change anymore, indicating convergence. Thus, the final centroids after convergence are:

- Centroid 1: $(-2.5, -3.5)$
- Centroid 2: $(4.5, 4.67)$
- Centroid 3: $(-7, -7)$

The points assigned to each centroid are:

x	y	Assigned Centroid
1	2	$(4.5, 4.67)$
3	3	$(4.5, 4.67)$
2	2	$(4.5, 4.67)$
8	6	$(4.5, 4.67)$
7	8	$(4.5, 4.67)$
6	7	$(4.5, 4.67)$
-3	-3	$(-2.5, -3.5)$
-2	-4	$(-2.5, -3.5)$
-7	-7	$(-7, -7)$

T7. Between the two starting set of points in the previous two questions, which one do you think is better? How would you measure the 'goodness' quality of a set of starting points?

Solution. To determine which set of starting points is better between the two previous questions, we can consider the following criteria for measuring the 'goodness' of a set of starting points:

- **Convergence Speed:** A better set of starting points should lead to faster convergence of the K-means algorithm. This can be measured by the number of iterations required to reach convergence.
- **Cluster Compactness:** The resulting clusters should be compact and well-separated. This can be measured using metrics such as the Within-Cluster Sum of Squares (WCSS) or Silhouette Score.
- **Stability:** A good set of starting points should lead to consistent clustering results across multiple runs of the algorithm. This can be evaluated by running the K-means algorithm multiple times with different random initializations and measuring the variance in the resulting clusters.
- **Interpretability:** The resulting clusters should be interpretable and meaningful in the context of the data. This can be assessed qualitatively by examining the characteristics of the data points within each cluster.

In the previous two questions, we can analyze the results based on these criteria. In Problem T05, the starting points led to convergence in **2 iterations**, while in Problem T06, it also converged in **2 iterations**.

However, the clusters formed in Problem T05 appear to be **more compact** and **well-separated** compared to those in Problem T06.

Therefore, based on the criteria mentioned above, we can conclude that:

$$\boxed{\textbf{the starting points in Problem T05 are better}}$$

OT2. What would be the best K for this question? Describe your reasoning.

Solution. To determine the best value of K for the given dataset, we can consider several approaches and reasoning:

- **Elbow Method:** This method involves plotting the Within-Cluster Sum of Squares (WCSS) against different values of K. The idea is to look for an 'elbow' point in the plot where the rate of decrease in WCSS starts to slow down. This point indicates a good balance between cluster compactness and the number of clusters.
- **Silhouette Score:** This metric measures how similar a data point is to its own cluster compared to other clusters. A higher silhouette score indicates better-defined clusters. We can calculate the silhouette score for different values of K and choose the one that maximizes the score.
- **Domain Knowledge:** If we have prior knowledge about the data, we can use that information to guide our choice of K. For example, if we know that the data represents three distinct categories, we might choose K=3.
- **Visual Inspection:** For low-dimensional data, we can visualize the clusters formed by different values of K and choose the one that appears to best capture the underlying structure of the data.

In the case of the given dataset, we can start by applying the Elbow Method and Silhouette Score to evaluate different values of K. Based on the visual inspection of the data points, it appears that there are three distinct clusters:

- A cluster around the points $(1, 2)$, $(2, 2)$, and $(3, 3)$.
- A cluster around the points $(6, 7)$, $(7, 8)$, and $(8, 6)$.
- A cluster around the points $(-3, -3)$, $(-2, -4)$, and $(-7, -7)$.

Therefore, based on the analysis and reasoning above, we can conclude that:

$$\boxed{\textbf{the best value of K is 3}}$$

3 My heart will go on

Note: Many parts of this exercise are adapted from Kaggle Python Tutorial on Machine Learning

In this part of the exercise we will work on the Titanic dataset provided by Kaggle. The Titanic dataset contains information of the passengers boarding the Titanic on its final voyage. We will work on predicting whether a given passenger will survive the trip.

Let's launch Jupyter and start coding!

We start by importing the data using Pandas.

```
train_url = "http://s3.amazonaws.com/assets.datacamp.com/
    course/Kaggle/train.csv"
train = pd.read_csv(train_url) #training set

test_url = "http://s3.amazonaws.com/assets.datacamp.com/course
    /Kaggle/test.csv"
test = pd.read_csv(test_url) #test set
```

Both train and test are dataframes. Use the function `train.head()` and `train.tail()` to explore the data. What do you see?

Use the function `describe()` to get a better understanding of the data. You can read the meaning of the data fields at https://www.kaggle.com/c/titanic/data.

Looking at the data, you will notice a lot of missing values. For example, some age is NaN. This is normal for real world data to have some missing values. There are several ways to handle missing values. The simplest is to throw away any rows that have missing values. However, this usually reduce the amount of training data you have. Another method is to guess what the missing value should be. The simplest guess is to use the Median or Mode of the data. For this exercise we will proceed with this.

T8. What is the median age of the training set? You can easily modify the age in the dataframe by:

```
train["Age"] = train["Age"].fillna(train["Age"].median())
```

Solution. The solution goes here.

Note that you need to modify the code above a bit to fill with `mode()` because `mode()` returns a series rather than a single value.

T9. Somefields like 'Embarked' are categorical. They need to be converted to numbers first. We will represent S with 0, C with 1, and Q with 2. What is the mode of Embarked? Fill the missing values with the mode. You can set the value of Embarked easily with the following command.

```
train.loc[train["Embarked"] == "S", "Embarked"] = 0
```

Solution. The solution goes here.

Do the same for Sex.

T10. Write a logistic regression classifier using gradient descent as learned in class. Use PClass, Sex, Age, and Embarked as input features. You can extract the features from Pandas to Numpy by:

```
data = np.array(train[["PClass","Sex","Age","Embarked"]].
    values)
```

Check the datatype of each values in data, does it make sense? You can force the data to be of any datatype by using the command:

```
data = np.array(train[["PClass","Sex","Age","Embarked"]].
    values, dtype = float)
```

Solution. The solution goes here.

When you evaluate the trained model on the test set, you will need to make a final decision. Since logistic regression outputs a score between 0 and 1, you will need to decide whether a score of 0.3 (or any other number) means the passenger survive or not. For now, we will say if the score is greater than or equal to 0.5, the passenger survives. If the score is lower than 0.5 the passenger will be dead. This process is often called 'Thresholding.' We will talk more about this process later in class.

To evaluate your results, we will use Kaggle. Kaggle is a website that hosts many machine learning competitions. Many companies put up their data as a problem for anyone to participate. If you are looking for a task for your course project, Kaggle might be a good place to start. You will need to make sure that your output is in line with the submission requirements of Kaggle: a csv file with exactly 418 entries and two columns: PassengerId and Survived. Then, use the code provided to make a new dataframe using `DataFrame()`, and create a csv file using `to_csv()` method from Pandas.

To submit your prediction, you must first sign-up for an account on kaggle.com. Click participate to the competition at https://www.kaggle.com/c/titanic then submit your csv file for the score.

The output file should have two columns: the passengerId and a 0,1 decision (0 for dead, 1 for survive). As shown below:

```
PassengerId,Survived
892,0
893,1
894,0
```

T11. Submit a screenshot of your submission (with the scores). Upload your code to courseville.

Solution. The solution goes here.

T12. Try adding some higher order features to your training (x_1^2, x_1x_2, ...). Does this model has better **accuracy on the training set**? How does it perform on the **test set**?

Solution. The solution goes here.

T13. What happens if you reduce the amount of features to just Sex and Age?

Solution. The solution goes here.

OT3. We want to show that matrix inversion yields the same answer as the gradient descent method. However, there is no closed form solution for logistic regression. Thus, we will use normal linear regression instead. Re-do the Titanic task as a regression problem by using linear regression. Use the gradient descent method.

Solution. The solution for the **OT** problem goes here.

OT4. Now try using matrix inversion instead. However, are the weights learned from the two methods similar? Report the Mean Squared Errors (MSE) of the difference between the two weights.

Solution. The solution for the **OT** problem goes here.

4 Fun with matrix algebra [OPTIONAL]

Prove the following statements. All of them can be solved by first expanding out the matrix notation as a combination of their elements, and then use the definitions of trace and matrix derivatives to help finish the proof. For example, the (i, j) element of $Y = AB$ is $Y_{i,j} = \sum_m A_{i,m}B_{m,j}$.

OT5. $\nabla_A tr\,(AB) = B^T$

Proof. Given the function $f(A) = tr(AB)$, we want to find the gradient of f with respect to the matrix A. First, we can express the trace of the product AB in terms of the elements of the matrices:

$$tr(AB) = \sum_i (AB)_{i,i} = \sum_i \sum_j A_{i,j}B_{j,i}$$

To find the gradient $\nabla_A f(A)$, we need to compute the partial derivative of f with respect to each element $A_{k,l}$:

$$\frac{\partial f}{\partial A_{k,l}} = \frac{\partial}{\partial A_{k,l}} \sum_i \sum_j A_{i,j}B_{j,i}$$

Since the only term in the double sum that depends on $A_{k,l}$ is when $i = k$ and $j = l$, we have:

$$\frac{\partial f}{\partial A_{k,l}} = B_{l,k}$$

Thus, the gradient $\nabla_A f(A)$ is given by the matrix whose (k, l) element is $B_{l,k}$, which is precisely **the transpose** of matrix B:

$$\nabla_A f(A) = B^T$$

Therefore, we have shown that:

$$\nabla_A tr(AB) = B^T$$

OT6. $\nabla_{A^T} f(A) = (\nabla_A f(A))^T$

Proof. Let $f(A)$ be a scalar function of the matrix A. To prove this, we start by expressing the elements of the gradients. The (i, j) element of the gradient $\nabla_A f(A)$ is given by:

$$(\nabla_A f(A))_{i,j} = \frac{\partial f}{\partial A_{i,j}}$$

Similarly, the (i, j) element of the gradient $\nabla_{A^T} f(A)$ is given by:

$$(\nabla_{A^T} f(A))_{i,j} = \frac{\partial f}{\partial (A^T)_{i,j}}$$

Since $(A^T)_{i,j} = A_{j,i}$, we can rewrite the partial derivative with respect to $(A^T)_{i,j}$ as:

$$(\nabla_{A^T} f(A))_{i,j} = \frac{\partial f}{\partial (A^T)_{i,j}} = \frac{\partial f}{\partial A_{j,i}}$$

Now, we can see that the (i, j) element of $\nabla_{A^T} f(A)$ corresponds to the (j, i) element of $\nabla_A f(A)$:

$$(\nabla_{A^T} f(A))_{i,j} = (\nabla_A f(A))_{j,i}$$

This shows that:

$$\nabla_{A^T} f(A) = (\nabla_A f(A))^T$$

OT7. $\nabla_A tr\left(ABA^T C\right) = CAB + C^T AB^T$

Hint: Try first solving the easier equation of $\nabla_A tr\left(BAC\right) = \left(CB\right)^T = B^T C^T$

Proof. Let $f(A) = tr(ABA^TC)$. We want to find the gradient of f with respect to the matrix A. First, we can express the trace in terms of the elements of the matrices:

$$tr(ABA^TC) = \sum_i (ABA^TC)_{i,i} = \sum_i \sum_j \sum_k \sum_l A_{i,j} B_{j,k} A^T_{k,l} C_{l,i}$$

Since $A^T_{k,l} = A_{l,k}$, we can rewrite the expression as:

$$tr(ABA^TC) = \sum_i \sum_j \sum_k \sum_l A_{i,j} B_{j,k} A_{l,k} C_{l,i}$$

To find the gradient $\nabla_A f(A)$, we need to compute the partial derivative of f with respect to each element $A_{p,q}$:

$$\frac{\partial f}{\partial A_{p,q}} = \frac{\partial}{\partial A_{p,q}} \sum_i \sum_j \sum_k \sum_l A_{i,j} B_{j,k} A_{l,k} C_{l,i}$$

The only terms in the triple sum that depend on $A_{p,q}$ are those where $i = p$ and $j = q$ or $l = p$ and $k = q$. Thus, we have:

$$\begin{aligned}
\frac{\partial f}{\partial A_{p,q}} &= \frac{\partial}{\partial A_{p,q}} \sum_i \sum_j \sum_k \sum_l A_{i,j} B_{j,k} A_{l,k} C_{l,i} \\
&= \frac{\partial}{\partial A_{p,q}} \left[\sum_k \sum_l A_{p,q} B_{q,k} A_{l,k} C_{l,p} + \sum_i \sum_j A_{i,j} B_{j,q} A_{p,q} C_{p,i} \right] \\
&= \frac{\partial}{\partial A_{p,q}} A_{p,q} \left[\sum_k \sum_l B_{q,k} A_{l,k} C_{l,p} + \sum_i \sum_j A_{i,j} B_{j,q} C_{p,i} \right] \\
&= \sum_k \sum_l B_{q,k} A_{l,k} C_{l,p} + \sum_i \sum_j A_{i,j} B_{j,q} C_{p,i} \\
&= \sum_k \sum_l B_{q,k} A^T_{k,l} C_{l,p} + \sum_i \sum_j C_{p,i} A_{i,j} B_{j,q} \\
&= (BA^TC)_{q,p} + (CAB)_{p,q} \\
&= (BA^TC)^T_{p,q} + (CAB)_{p,q} \\
&= (C^TAB^T + CAB)_{p,q} \\
\frac{\partial f}{\partial A_{p,q}} &= (CAB + C^TAB^T)_{p,q}
\end{aligned}$$

Thus, the gradient $\nabla_A f(A)$ is given by the matrix whose (p, q) element is $(CAB + C^TAB^T)_{p,q}$:

$$\nabla_A f(A) = CAB + C^TAB^T$$