Assignment 5 DATA 301

1)

	Age	Income	Owns Car	distance
-	1	2	Yes	1
	3	3	Yes	$\sqrt{2}$
	1	1	No	$\sqrt{2}$
	2	3	Yes	1
	1	3	No	$\sqrt{2}$
-	3	2	No	1
-	2	1	Yes	1
	1	1	Yes	$\sqrt{2}$
-	3	2	No	1
	3	3	Yes	$\sqrt{2}$

a person whom is middle age, middle class owns a car
(the rows w/ the dashes are the ones I chose)

2) iteration 1 5

	age	salary	C1=(23,22)	C2=(40,80)	C3=(70,30)
1	23	22	0	60.44	47.67
2	33	50	29.73	30.8	33.6
3	40	80	60.44	0	58.3
4	11	5	20.8	80.41	64.07
5	70	30	47.67	58.3	0

C1 : [1, 2, 4]
C2 : [3]
C3 : [5]

iteration 2

	age	salary	C1=(22.3, 25.7)	C2=(40,80)	C3=(70,30)
1	23	22	3.76	60.44	47.67
2	33	50	26.55	30.8	33.6
3	40	80	57.11	0	58.3
4	11	5	23.58	80.41	64.07
5	70	30	47.49	58.3	0

C1 : [1, 2, 4]
C2 : [3]
C3 : [5]

3)

- Can be difficult to predict the number of clusters

- The initial centroids may have a significant impact on the results. That may cause the solution generated from K-means to be a local optimum.

4) Using a neural network to predict words in speech recognition software

5) With regression the output is a real number whereas with Classification the output is either True/False or a choice out of a list of choices

6.) Supervised learning has both testing and training sets whereas unsupervised learning has NO training data.

7.) This is useful because sometimes the data doesn't fit nicely to a y=mx+b equation. For instance, the data could follow a parabola. Using a polynomial transformation fixes this issue.

8.) model.fit takes in 2 parameters → the training data's x-values and the training data's values. The x values must be a vector.

```
X = np.array([1, 2, 5, 7, 10])
y = np.array([2, 4, 6, 8, 13])

model = LinearRegression(fit_intercept = True)
model.fit(X[:, np.newaxis], y)
```

9.) make_pipeline sequentially applies a list of transformations, or in other words, it pipelines a bunch of steps that modify the input.

```
X=np.array([1,2,5,7,10])
y=np.array([2,4,6,8,15])

model = make_pipeline(PolynomialFeatures(degree=4),
LinearRegression())

model.fit(X[:, np.newaxis], y)
```

10.) model.predict uses the model and the testing set's x values to predict the testing set's y values. In other words, this allows us to predict the future. This should be called after model.fit.

11.) The add function allows us to add a layer of nodes to the neural network. The compile function configures the learning process(ie. tells if we need to train a binary classification or multi-class classification problem). Also, we provide a loss function(what we're trying to minimize) and a list of metrics.

12) model.evaluate computes the loss, along with the other metrics passed to the compile function of the test data.

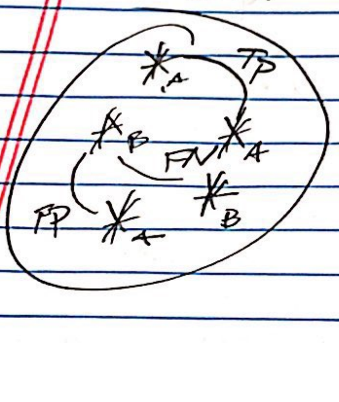
13) You could use the f1 score. The f1 score = $2 * \frac{\text{the precision} * \text{recall}}{\text{precision} + \text{recall}}$.

precision = $\frac{\#\text{true pos}}{\#\text{true pos} + \#\text{false pos}}$

recall = $\frac{\text{true pos}}{\text{true pos} + \text{false neg}}$

13 (cont) This is a good metric b/c it takes into account each type of possible outcome in its classification.

CA



14) You could start k at 0 and while you don't ~~g~~ obtain a decent grouping, increment k and run the algorithm again.