Question 1

Page A : Probability of clicking on button = 20/100 = 0.2 = 20%

Page B: Probability of clicking on button = 15/70 = 0.2142 = 21.42 %

Page B had a better probability of generating a click however this may not be statistically significant and may be purely due to chance. Lets look at it in more details……

Assumptions:
Its hard to make any other statistical inference from just two point estimates. So we will change and assume that these are actually the means after a number of tests. Thus after a number of tests on page A and page B it was observed that the mean conversion rate was 20% for Page A and 21.42% for Page B. We will assume that the distributions are normally distributed and the variance for both Pages is 10%.

Now we need to figure out if the the difference in Means is statistically significant.
Let $X = U_a - U_b$ (difference in population means for Page A and Page B)

Null hypothesis X = 0
Alternative hypothesis X != 0

The test statistic then is:

$$t = \frac{(P_a - P_b) - (U_a - U_b)}{(S_p^2/n_1 + S_p^2/n_2)^{1/2}}$$

Where $$S_p = \frac{(n_1 - 1)S_1^2 + (n_1 - 1)S_2^2}{n_1 + n_2 - 2}$$

Thus

$$S_p = \frac{(100 - 1)10^2 + (70 - 1)10^2}{100 + 70 - 2} = \frac{16800}{168} = 100$$

$$t = \frac{(20 - 21.42) - 0}{(168^2/100 + 168^2/70)^{1/2}} = \frac{-1.42}{(282.4 + 403.2)^{1/2}} = -1.42/26.18 = -0.05423$$

The p-value at 90% significance is +/- 1.653. Thus we are not able to reject the null.

Thus we cannot say if the Page A and Page B have any statistically significant differences.

Question 2

```
import json
import datetime

## Data input
## Assume this reads in the Json data. Will need to change depending on format of
the json file.
with open('data.json') as data_file:
    data = json.load(data_file)

## Assume data has the json information with keys as user_id, timestamp, tweet
## Assume we have a corpus of words that we deem to be significant. These
may be particular words around a topic that we want to find similarity for.
For example if we are trying to cluster user_id based on NFL team
affiliation we may have football terms like team names, stadium names, NFL
player names, nfl terms like touchdown etc. If there is no particular topic
that you want to cluster around then there are readymade general corpuses
that we can use.

Preprocessing:

1) Change data['tweet'] into bag of words after stemming.
2) Keep only words that appear in  corpus about as most relevant.
3) Next create categorical variables from this data['tweet'] column (Note: this
will blow up the key space of the data many orders of magnitude)
4) Change data['timestamp'] to distinct classes/groups for a timeframe, for
example each 10 minutes say (i.e tweets from 10:30 to 10:40 get a single class)

Training:

1) Consider the bag of words (and the categorical variables from it) and the
timestamp as the feature set. We may either combine all the bag of words for a
user_id (option 1: each user_id will "predominantly" be assigned to one cluster)
or keep the bag of words separate (option 2: each tweet will "predominantly" be
assigned to one cluster, while a user_id may be assigned to many clusters)

2) Consider the user_id as the label.
3) Run clustering algorithms. We will use a Gaussian Mixture Model clustering
algorithm here because we don't have an algorithm capable of making clear topic
distinction like humans do. K-means works great if tweets fall distinctly into
different topics and the algorithm can understand it. Absent that, it is better to
let the algorithm allow probabilistic outcomes to each tweet so that each tweet
may be part of many clusters until we have enough learning done.


Testing:

To find out more about user_id (e.g.: what team affiliation etc) we just look up
```

```
the user_id (label) and find the cluster that it has been most assigned to. If we
used option 1 above we are done (but it may be the case that the clusters are
sparse) else if we used option 2 above the user_id will be in multiple clusters
and we may have to do some post processing (i.e the cluster to which the user_id
is assigned to most times etc. For example football tweets will be clustered with
other football tweets, home improvement tweets with other home improvement tweets,
kids schools tweets with other kids schools tweets etc.)
```

Question 3

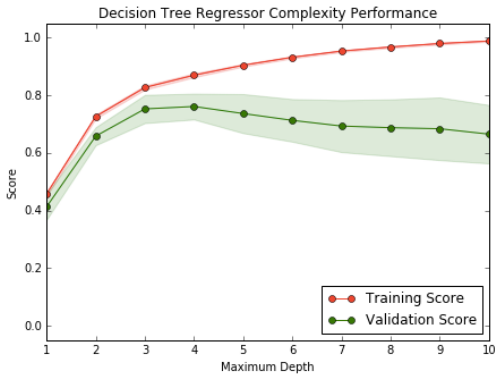
Overfitting occurs when a model is allowed to become more complex than the data suggests it should be. For example if we have data that is mostly linear but we get a model that is quadratic. This happens because the model tried to fit/explain every data point it was given. A good model would consider the possibility of noise and not to explain every point. There are a few ways to detect and prevent over fitting:

Detection:

Overfitting happens when the model cannot generalize. It is overly complicated for the data. It can explain the data it has seen very well but cannot explain unseen data well. Thus we will see a very low prediction error on the training set but a very high prediction error on the testing set.

In general models are trained using training data and tested using validation data to prevent overfitting (test data is held out and seen by the model only once.) The prediction error on the training data and the validation is noted for each attribute selection of the model (for example max_depth in a DecisionTree.) The attribute for which both training and validation data show decrease in prediction is elected.

In the figure below both training and validation score improve as the model becomes more complicated and then at around max_depth = 3 the model is too complicated. The training score keeps on improving but the validation score decreases (i.e it has overfit)



Prevention:

1) We can reduce the chances of over fitting by regularization. Regularization prevents the model from becoming overly complicated and the coefficients from becoming too large. This makes the model less

sensitive to spurious changes in data (noise.) e.g: Regularized logistic regression.

2) We can prevent the model in a classification from becoming too complex by either keeping a cap on the max_depth (for a tree for example) or by letting the tree to grow as much as it needs and then tree pruning to reduce the depth.

3) We can also make sure that the model generalizes well by using Stratified Sampling. This ensures that the training/testing set see the output labels in the same proportion that they are in the population.

4) We can remove outliers from the training set to begin with. If there are no outliers the model is likely to overfit to it. Also we should make sure all the features are scaled to be in the same range. This will keep the model well behaved when training.

5) Finally we can use k-fold cross validation (especially if we have limited data) to make sure that the model gets to see and train on all the data as well as tested on all the data. The error estimate for the model is the average estimate for each fold. Its hard to explain why this works. It is counter intuitive but it works.

Question 4:

We need to gather data from the user interaction with the tool and then learn the outcomes from those user interaction so that we can create those outcomes automatically. For example, for each screen that the user is on we log the following data:

Current window elected, Current object displayed, cursor location, mouse button clicks, keyboard inputs, number of clicks.

Then we break the data we gathered into

1) Input observations/features: Current window elected, Current object displayed, cursor location, mouse button clicks, keyboard inputs

2) Output label: Menu item user selects for each screen

We then train a model using these features and labels. We can either use a classification model or a clustering model. We don't need data preprocessing here since the inputs are actions and outputs are actions as well so we don't expect outliers. We go through the training/testing phases multiple times till we have a working model.

Next after we have a working model, we predict the context sensitive menu for each screen based on the input features/data we have observed form the user till then. Thus if the user is on a screen displaying a surface, the mouse cursor is near the surface and we detect a right mouse click then output label should be "smooth surface" which we can include in the context sensitive menu.

Question 5:

Regularization is used to keep a cap on the size of the coefficients of the features. If the coefficients of the features become too big, small changes in input data would cause large changes in the output of the model, thus making it susceptible to noise and outliers. If coefficients are unbalanced, that is some are large and some are small it means some features have uneven influence on the output label. Thus

the the model has over fit to those features. Having regularized coefficients prevents this and stops the model from over fitting to each outlier/point.

There is no point regularizing if:

1) Your preprocessing shows that there are no outliers in the data
2) All the features are scaled to have similar ranges.

If we take regularization too far and end up making all the coefficients too small, the model will lose its predictive power and will end up outputting a close to constant value for any input.

Question 6

This is a product recommendation algorithm. There are many algorithms for this.

Here is the classic user-user collaborative filtering algorithm. Here are the steps:

1) Since we already have a purchase history for each customer we know what each customer bought. Thus each customer has a bag of items they bought.

2) This represents a vector in multidimensional space (i.e each customer is a vector.) For example we know customer A bought 10 diapers, 3 beers and 4 coffee pods in their previous purchases. We have a vector for customer A that is as long as all the products sold by the store. Each dimension for the vector is a product and A will get 10, 3, and 4 along dimensions that represent diapers, beer and coffee respectively.

3) We create vectors for all customers in the same way as in #2 above.

4) Next we find all the customers that are close to customer A using angle between the vectors.

$$\text{E.g} \quad \frac{B \cdot A}{|B| * |A|}$$

will give the cosine of the angel between the vectors. Smaller this quantity the more close A and B are in the purchasing behavior.

5) Find all the products that Customer B bought but Customer A did not. Lets call them x, y, z. Next time when Customer A comes shopping print them out coupons for x, y, z with a unique coupon number.

We can measure if a coupon was used based on the unique coupon numbers.

$$\text{How well is the system working:} = \frac{\text{\# of coupons redeemed}}{\text{\# of coupons printed out.}}$$

There are other techniques of course:

1) We can use item-item collaborative filtering.
2) We can use clustering to cluster the vectors together.

Question 7

Lets pick google. Google is an interesting company pushing boundaries. After it started working on autonomous cars bunch of other companies started working on the same issues. Thus google is an extremely innovative company. If I was hired as a machine learning engineer at google I could see working on the autonomous car program and in one year moving to newer challenges. Here are some other challenges the can be tackled:

1) Today the web is still largely static. Users are tracked across the web using "third party cookies" but these are predominately used only to serve Ads. Each webpage is still static and not customized for each user (the only customization are the Ads.) I think this is an area that could use change. For example a news site could bubble up news stories based on your previous browsing history, a product site like amazon can bubble up products based on browsing history etc.

2) My capstone project was predicting airline delays from past data. By reducing outliers so that we ask the model to only make predictions within a small time window the prediction accuracy is increased and we can make predictions in advance. I think model like this can be part of the google flights. When google shows details of flight a customer is covering purchasing, it can predict the flight delays using the model and show these predicted flight delays as one of the details in addition to fare, flying time etc. The project can be a learning model that learns everyday on previous X months of rolling data so its predictions are current.

3) I think investment management is ripe for disruption as well. Most of the mutual funds are controlled by individuals who make investment choices using objective research (which can easily be programmed) and gut instincts (for which we need machine learning.) There shouldn't be any reason investment management couldn't also be automated.

Anyway I would like to continue working on interesting, new, novel challenges.