notes

July 17, 2024

1 Phase 4 Notes

2 Distance Metrics

2.1 KNN

KNN is an effective classification and regression algorithm that uses nearby points in order to generate a prediction.

1. Choose a point
2. Find the K-nearest points
    1. K is a predefined user constant such as 1, 3, 5, or 11
3. Predict a label for the current point:
    1. Classification - Take the most common class of the k neighbors
    2. Regression - Take the average target metric of the k neighbors
    3. Both classification or regression can also be modified to use weighted averages based on the distance of the neighbors
4. Don't technically train or fit
5. Efficient on small-mid size data not good for large data

2.1.1 Assumptions of Distance Based Classifiers

distance helps us quantify similarity

2.1.2 Manhattan distance

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

[ ]:
```
# Locations of two points A and B
A = (1, 7, 12)
B = (-1, 0, -5)

manhattan_distance = 0

# Use a for loop to iterate over each element
for i in range(3):
    # Calculate the absolute difference and add it
    manhattan_distance += abs(A[i] - B[i])
```

```
manhattan_distance
```

[ ]: 26

2.1.3 Euclidean distance

$a^2 + b^2 = c^2$, or the **Pythagorean theorem**!

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

[ ]:
```
from math import sqrt

# Locations of two points A and B
A = (1, 7, 12)
B = (-1, 0, -5)

euclidean_distance = 0

# Use a for loop to iterate over each element
for i in range(3):
    # Calculate the difference, square, and add it
    euclidean_distance += (A[i] - B[i]) ** 2

# Square root of the final result
euclidean_distance = sqrt(euclidean_distance)

euclidean_distance
```

[ ]: 18.49324200890693

2.1.4 Minkowski distance

A Normed Vector Space is just a fancy way of saying a collection of space where each point has been run through a function. It can be any function, as long it meets two criteria: 1. the zero vector (just a vector filled with zeros) will output a length of 0, and 2. every other vector must have a positive length

Both the Manhattan and Euclidean distances are actually *special cases of Minkowski distance*. Take a look:

$$d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^c\right)^{\frac{1}{c}}$$

2.1.5 Hamming Distance

Hamming distance can even be used to compare strings

2.1.6 How adjusting K works

2.1.7 Big O is Exponential for KNN

Note that KNN isn't the best choice for extremely large datasets, and/or models with high dimensionality. This is because the time complexity (what computer scientists call "Big O", which you saw briefly earlier) of this algorithm is exponential.

2.1.8 Best value for K

arrived at through testing on data set and trying diff values

2.2 Lecutre on KNN

- Pick K for low bias low variance
- Fitting doesn't train, it just stores the locations in the feature space. What's the distance, get the closest distance.
- Hyper tuning the number of neighbors we have
- Low K = overfit, High K = underfit
- Must scale the features!
- Kfolds, GridSearchCV etc standardize after splitting
- next(fold_index) will show the iteration of indexes in cross validation
- cross validation finding the best score
- lower k that predicts better is usually better
- weighted averages: multiply support by
- hidden dimensions latatent space
- predicting about generalizing well
- KNN is a lazy algorithm it works well with smaller data sets
  - over 100K it starts to be too big
  - columns matter too
- Alternative to OHE? Encode one column with all the values
- More features = more dimensions = more sparsity
  - makes it harder to train or predict and can overfit
  - volume scales exponentially
  - affects all algorithms
  - more columns can capture variance but you can over do it
- Feature spaces
  - cosine used for recommendations
  - hamming mlp, distance between words

3 Lloyd's vs Fair Lloyds

K clustering fair lloyd's attempts to make cost between clusters fair by defining demographics groups where costs should be compared and altering clustering based on that, small increase in cpu.

4 GridSearchCV

Cross validation and hyper parameter tuning all in one It's exhaustive and how good it is depends on what params you feed it, it can waste a lot of time for no gain if not done thoughtfully.

5 Pickle

serialize state and read or write it to a file

6 Machine Learning Pipelines

helps avoid data leakage and lets you make a workflow

[ ]:
```
from sklearn.pipeline import Pipeline

# Create the pipeline
pipe = Pipeline([('mms', MinMaxScaler()),
                 ('tree', DecisionTreeClassifier(random_state=123))])

# Create the grid parameter
grid = [{'tree__max_depth': [None, 2, 6, 10],
         'tree__min_samples_split': [5, 10]}]


# Create the grid, with "pipe" as the estimator
gridsearch = GridSearchCV(estimator=pipe,
                          param_grid=grid,
                          scoring='accuracy',
                          cv=5)

# Fit using grid search
gridsearch.fit(X_train, y_train)

# Calculate the test score
gridsearch.score(X_test, y_test)
```

6.1 Used with other libraries

Cross validate accepts a param for a pipeline and possibly others so it's well integrated with some libraries.

7 Lecture on Pipelines and GridSearchCV

- Hyperparameters exist for both parametric and non parametric models
- Pipeline solves
  - K Fold cross validation takes loops and can get unwieldly
  - crossval for each fold

  - streamline this preprocessing
  - do things in parallel
- Pipeline takes
  - constructor takes in a list of tuples as steps
    - user label and transformer/estimator
  - pipeline.fit
  - pipeline.transform
- GridSearchCV
  - pipelinename___hyperparameter
  - .best_estimator_
  - refit on entire train after for better predictions
  - ending in _ means it was filled after the fitting
- (add the rest of the lecture)

8 Ensemble

Model that uses more than one model to make a prediction. They often aggregate results. Usually used in supervised learning.

They are resilient to variance, think a group of specialists all weighing in on something to come up with wisdom of the crowd.

Over and under estimates cancel out which is called smoothing.

9 Bootstrap Aggregation

Bagging, which is short for ***Bootstrap Aggregation*** is two ideas bootstrap resampling and aggregation.

Bootstrap resampling is a statistical method used to estimate the distribution of a statistic (e.g., mean, variance) by sampling with replacement from the original dataset.

Sampling with Replacement Sampling with replacement means that when selecting elements from a dataset, each element is returned to the dataset after being selected. This allows the same element to be chosen multiple times in the sampling process.

Aggregation is combining. In this case it is combining the bootstrap samples.

The process for training an ensemble through bootstrap aggregation is as follows:

1. Grab a sizable sample from your dataset, with replacement
2. Train a classifier on this sample
3. Repeat until all classifiers have been trained on their own sample from the dataset
4. When making a prediction, have each classifier in the ensemble make a prediction
5. Aggregate all predictions from all classifiers into a single prediction, using the method of your choice

Decision Trees are often used because they are sensitive to variance but they don't have to be used.

10 Random Forest

Ensemble of decision trees, but decision trees use a greedy algorithm that maximizes information gain at each step. We need each tree to be different. **Bagging** and **subspace sampling** let the trees have more variance.

For each tree in the dataset:

1. Bag 2/3 of the overall data – in our example, 2000 rows
2. Randomly select a set number of features to use for training each node within this – in this example, 6 features

3. Train the tree on the modified dataset, which is now a DataFrame consisting of 2000 rows and 6 columns

4. Drop the unused columns from step 3 from the out-of-bag rows that weren't bagged in step 1, and then use this as an internal testing set to calculate the out-of-bag error for this particular tree

- Great for large complex datasets
- Not prone to overfitting
- Data doesn't need to be standardized
- Uses bootstrap sampling to randomly select different samples
- Uses random feature selections
- Can fail to capture linear relationships
- Smooths out classes so it's not as subject to influence by single points

10.1 Bagging for Random Forest

1. obtain a portion of the data with replacement
2. use this data to build a tree
3. remaining data is **Out-of-Bag Data** or **OOB**.

4. OOB is used as test set to calculate the **Out-Of-Bag Error** to estimate performance.

11 Subspace Sampling for Random Forest

Further increases variability between trees by using a subset of features for each tree.

11.1 Random Foreset Visual of Algorithm

11.2 Resilient to overfitting

due to the number of trees and their variance it is resilient to overfitting. Finds signal in the noise.

Each tree "votes" on the overall outcome.

11.3 Benefits

Strong Performance - it is an ensemble method so and it tends to outperform many models.

Interpretability - it is called a **glass box model** because it is transparent and easy to see how it arrived at a solution.

11.4 Drawbacks

Computational Cost - It can be slow to train on large data sets.

Memory Footprint - It has to store all the data for each tree which can end up being hundreds of MBs. Logistic regression only needs to store the coefficients.

11.5 Random Forest Paper and Website

- Random forests paper
- Random forests website

12 Gradient Boosting and Weak Learners

12.1 Weak Learners

A model that is only good at predicting slightly better than random chance

1. Train a single weak learner
2. Figure out which examples the weak learner got wrong
3. Build another weak learner that focuses on the areas the first weak learner got wrong
4. Continue this process until a predetermined stopping condition is met, such as until a set number of weak learners have been created, or the model's performance has plateaued

12.2 Boosting vs Random Forest

Very similar to random forests: ensembles of high variance models that aggregate to make a prediction. Both often use tree models, boosting can use other models though. |Boosting|Random Forest| |———|————-| |Iterate|Parallel| |Corrects on Prior Trees|Trees don't know of each other| |Ensemble of Weak Learners|Ensemble of Strong Learners| |Very Resistant To Overfitting|Resistant to Overfitting| |Weighted Votes|Simple Votes| |Weight on Trees That Solve Harder Problems|All Even Weights| |Aggregate Solves Easy Problems|No Interaction Like this|

13 AdaBoost

- One of the first boosting algorithms
- Uses weights on the sampling to increase weights on samples that the learner gets wrong, these weights increasing means the sample is more likely to end up in the bag
- Ensemble can guess easy on easy problems so they are given less weight

14 Gradient Boosted Trees

- Makes use of Gradient Descent

- Uses weak learners
- This is where it diverges from AdaBoost: It calculated the residuals next to see how far it is off
- Residuals are combined with a loss function
- Loss function is differentiable
- Loss function is inflated more where the model is more wrong, thus it will be pushed towards making a model focusing on these harder problems

→ How does gradient boosting work for a classification problem? How do we even make sense of the notion of a gradient in that context? The short answer is that we appeal to the probabilities associated with the predictions for the various classes. See more on this topic here. → Why is this called "*gradient* boosting"? Because using a model's residuals to build a new model is using information about the derivative of that model's loss function. See more on this topic here.

14.0.1 Learning Rate

γ – this is the greek letter, ***gamma*** which is for learning rate

Remember that too high of a learning rate is good to quickly train but won't find the best setting, and can lead to bouncing.

A small learning rate will take longer to train and can get stuck in local minimums easier but will find a better value

14.0.2 Algorithm

Use mean squared error (MSE) and want to minimize that <– done by gradient descent

Use the residuals (pattern in the residuals) to create an even better model

1. Fit a model to the data, $F_1(x) = y$
2. Fit a model to the residuals, $h_1(x) = y - F_1(x)$
3. Create a new model, $F_2(x) = F_1(x) + h_1(x)$
4. Repeat

Example of Iterative Steps https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoo

Parts adapted from https://github.com/ageron/handson-ml/blob/master/07_ensemble_learning_and_random_forests.ipynb

15 XGBoost - Extreme Gradient Boosting

- Handles missing values for you
- Runs on multiple cpu cores in parallel
- Distributes training across computer clusters
- Go-to competition Algorithm
- Always use multiple algorithms but it's a top dog right now

16 Recommendation Systems

- Allows predicting the future preference list

16.1 Matrix Factorization

- Singular Value Decomposition (SVD) and Alternating Least Squares (ALS)

16.2 Surprise Library

- Used to create recommendation systems and runs really optimally

16.3 Goal: Expose People to What They Like

- Predicts the future preference of a set of items or user
- Taps into the "long tail", there's very common items everyone buys but the long tail specific items, like a certain genre of music or special toy are long tail

16.4 Formal Definition

Recommendation Systems are software agents that elicit the interests and preferences of individual consumers [...] and make recommendations accordingly. They have the potential to support and improve the quality of the decisions consumers make while searching for and selecting products online.

16.5 Applications of Recommendation Systems

- Suggest items to a customer
- Estimate profit & loss of many competing items and make recommendations to the customer (e.g. buying and selling stocks)
- Recommend a product or service based on experience of the custoemr
- Show offers appealing to a customer

16.6 Types of Recommendation Systems

- Unpersonalized and Personalized

16.6.1 Unpersonalized

- EX: Youtube recommending the most viewed videos.

16.7 Personalized

Given: The profile of the "active" user and possibly some situational context, i.e. user browsing a product or making a purchase etc.

Required: Creating a set of items, and a score for each recommendable item in that set

Profile:

User profile may contain past purchases, ratings in either implicit or explicit form, demographics and interest scores for item features

> There are two ways to gather such data. The first method is to ask for explicit ratings from a user, typically on a concrete rating scale (such as rating a movie from one to five stars). The second is to gather data implicitly as the user is in the domain of the system - that is, to log the actions of a user on the site.

Each of these techniques make use of different similarity metrics to determine how "similar" items are to one another. * Euclidean distance * cosine similarity * Pearson correlation * Jaccard index (useful with binary data)

16.7.1 Content-Based Recommenders

Main Idea: If you like an item, you will also like "similar" items.

- These systems are based on the characteristics of the items themselves. "Try other items like this"
- Gives the user a bit more information on why they are seeing the recommendation
- Require manual or semi-manual tagging of products
- advanced systems can average all items a user liked

16.7.2 Collaborative Filtering Systems

Main Idea: If user A likes items 5, 6, 7, and 8 and user B likes items 5, 6, and 7, then it is highly likely that user B will also like item 8.

The key idea behind collaborative filtering is that similar users share similar interests and that users tend to like items that are similar to one another.

- Often based off user reviews
- Have a cold start problem on how to recommend things to new users that have no activity yet.

16.8 Utility Matrix reprsents the associated opinion that a user holds.

	Toy Story	Cinderella	Little Mermaid	Lion King
Matt		2		5
Lore	2		4	
Mike		5	3	2
Forest	5		1	
Taylor	1	5		2

$r_{\text{Mike,Little Mermaid}} = 3$.

A recommendation system tries to fill in the blanks. Most of the time these values are largely empty. The matrix above is what is known as an explicit rating. Each person has rated what they've seen. However we can infer or use judgement to determine how to use data for a recommendation system.

	Toy Story	Cinderella	Little Mermaid	Lion King
Matt		1		1
Lore	1		1	
Mike		1	1	1
Forest	1		1	
Taylor	1	1		1

These are **implicit** ratings because we are assuming that because a person has bought something, they would like to buy other items like it. Of course, this is not necessarily true, but it's better than nothing!

17 Clustering

Create clusters that have high similarity between the data belonging to one cluster while aiming for minimal similarity between clusters

17.1 K-Means Clustering

K determines the number of clusters and the algorithm optimizes around that

17.2 Hierarchial Agglomerative Clustering

You start with n clusters equal the number of data points and at each step you join two clusters. You stop joining when a certain criterion is reached.

17.3 Semi-Supervised Learning

Combine both concepts of supervised and unsupervised learning. Increasingly popular.

17.4 Market Segmentation with Clustering

Common and useful, we'll practice with a market segmentation dataset.

18 K-means Clustering

The most popular and widely used clustering algorithm, and clustering are one of the most popular unsupervised machine learning algorithms.

18.1 Goal

Intra-class similarity is high

Inter-class similarity is low

Similarity is determined by distance. Closer is more similar. * **Agglomerative hierarchical** algorithm starts with n clusters * **Non-heirarchical** chooses k initial clusters

Unsupervised and you do not know how many clusters you are looking for

18.2 Non-Hierarchical Clustering with K-Means Clustering

18.2.1 Process

1. Select k initial seeds
2. Assign each observation to the cluster to which it is "closest"
3. Loop
    - Cluster center is the mean of all points in the cluster, recalculated each iteration.
    - Each iteration reassign points to be part of the closest cluster center.

- Stop if there is no reallocation

18.2.2 Scikit-learn

```
[ ]: from sklearn.cluster import KMeans

# Set number of clusters at initialization time
k_means = KMeans(n_clusters=3)

# Run the clustering algorithm
k_means.fit(some_df)

# Generate cluster index values for each row
cluster_assignments = k_means.predict(some_df)

# Cluster predictions for each point are also stored in k_means.labels_
```

18.2.3 Evaluation with Variance Ratio

- Accepted metric in wide use is ***Variance Ratio*** aka ***Calinski Harabasz Score***
  - The variance of the points within a cluster to the variance of a point to points in other clusters.
  - We want intra-cluster variance to be low suggesting the clusters are tightly knit.
  - We want inter-cluster variance to be high suggesting that there is little to no ambiguity about which cluster a point belongs to.

Calculating Variance Ratio

```
[ ]: # This code builds on the previous example
from sklearn.metrics import calinski_harabasz_score

# Note that we could also pass in k_means.labels_ instead of cluster_assignments
print(calinski_harabasz_score(some_df, cluster_assignments))
```

18.2.4 Other Metrics

- Silhouette Score
- No metric is best, each have diff strengths weaknesses based on given goals.

18.2.5 Optimal K Value

1. Fit different K-means clustering objects for every k we want to try then compare the variance ratio scores of each.
2. Visualize results with an ***Elbow Plot*** - plots that we can easily see wehre we hit a point of diminishign returns. They are used with more than just variance ratios, one example is distortion another clustering metric.

Understanding the Elbow A note on elbow plots: higher scores aren't always better. Higher values of k mean introducing more overall complexity – we will sometimes see elbow plots that look like this:

$k = 20$ is technically better as a score but $k = 4$ is better because it balances model complexity with score

19 Hierarchical Agglomerative Clustering

- K-means uses Expectation-Maximization after we tell it to give us k clusters, however it can not have subgroups within subgroups
- Agglomerative Clustering to the rescue! It can have subgroups within subgroups
- It starts with n clusters with n = the number of data points then merges until some stopping criterion

19.1 Linking Clusters Together

- **ward** - merges two cluster on the least variance between them. Leads to more equally sized clusters
- **average** - merges the two clusters that have the smallest average distance between all points
- **complete** - merges the two clusters that have the smallest maximum distance between their points

Can affect the performance, which to use is based on the data and goals.

The following diagram demonstrates the clusters created at each step for a dataset of 16 points. Take a look at the diagram and see if you can figure out what the algorithm is doing at each step as it merges clusters together:

As you can see it takes the closest clusters and merges them into a single cluster. Below shows as the dots disappear the visualization is repalcing them with the newly calculated center.

19.1.1 Dendrograms and Clustergrams

- Easily visualize the results at any given step
- The image to the right above in the gif is a Dendrogram
  - shows the hierarchical realtionship between the various clusters that are computed throughout each step.

- The image below is a Clustergram
  - Visualize the same information by drawing lines representing each cluster at the each step

19.1.2 Use Cases

- market segmentation
  - things like market segmentation
- gain a deeper understanding of a dataset through cluster analysis
- photo sorting on smartphones

20 Common Problems with Clustering Algorithms

- No way of verifying the results are correct or not
  - Never treat results of a cluster as ground-truth ## Advantages and Disadvantages of K-Means Clustering ### Advantages
- Easy to implement
- Usually faster than HAC with reasonably small k and many features
- Objects can shift clusters
- Tighter clusters than HAC

20.0.1 Disadvantages

- Need the right value for k
- Scaling completely changes the results
- Starting points have a strong impact on final results, as seen below. Bad init is less likely than good init and you can run it multiple times.

20.1 Advantages & Disadvantages of HAC

20.1.1 Advantages

- ordered relationship between clusters, which can be useful when visualized
- smaller clusters which allows more granular understanding

20.1.2 Disadvantages

- Results depend on distance metric used
- Objects can be grouped badly early on and no way to move them
- We can't check visuals on more than 3 dimensions so it's hard to know when the algorithm was correct
- Clustergram below

21 Semi-Supervised Learning and Look-Alike Models

- Combining both to solve real world problems

21.1 Case 1: Look-Alike Models

- Find a similar audience
- Identify more customers/market segments that we can plausibly assume are equally valuable due to their similarity with valuable custmoers or market segement we already identified.
- Divide into two groups: the ones we know are valuable and everyone else
- Uses distance metric of choice to rate similarity of the unknown customers with the ones we have identified
- Once we know they are somewhat similar to the valuable group we can spend resources to capture them
- Likely see customers that are only somewhat similar to our valuable group
- Customer sthat look nothing like our known valuable customer segment
- It is a lot like clustering

- referred to as prospecting.
- choose resources to market to the customers that look like our valuabel customers to increase our top-of-funnel, meaning an increase to the number of potential customers that haven't shown interest in our product or cmpany yet but are likely to.

21.2 Use Case 2: Semi-Supervised Learning

- Known as weakly supervised learning too
- Generate Pseudo-labels that are possibly correct.
  - doesn't use clustering, it uses supervised learning algorithms in an unsupervised way.

21.2.1 Steps

1. **Train your model on your labeled training data**
2. **Use your trained model to generate pseudo-labels for unlabeled data**
3. **Combine the pseudo-labels with your actual data**
4. **Retrain your model on the new data set** #### Benefits

- It is risky
- When done well it can increase overall model performance by opening up access to much more data
- Saves a ton of money on labeling costs!

Downsides

- When data is really noisy incorrect lables will skew the model
- Feedback Loops
- More complicated problems tend to work less

Use a Holdout Set to Test

- As usual but even more important in this case, make sure to use ground-truth or non pseudo-code labels to test with.

22 Silhouette Coefficient

The Silhouette Coefficient is a measure used to evaluate the quality of clusters created by a clustering algorithm. It takes into account both the cohesion within clusters and the separation between clusters.

22.1 Definition

For a given data point i, the Silhouette Coefficient $s(i)$ is defined as:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

where: - $a(i)$ is the mean distance between i and all other points in the same cluster. - $b(i)$ is the mean distance between i and all points in the nearest cluster (the cluster with the smallest mean distance to i).

22.2 Interpretation

- $s(i)$ ranges from -1 to 1.
  - $s(i) \approx 1$: The data point is well-matched to its own cluster and poorly matched to neighboring clusters.
  - $s(i) \approx 0$: The data point is on or very close to the decision boundary between two neighboring clusters.
  - $s(i) \approx -1$: The data point is poorly matched to its own cluster and well-matched to a neighboring cluster.

22.3 Overall Silhouette Score

The overall Silhouette Score for a clustering is the mean Silhouette Coefficient of all data points:

$$S = \frac{1}{N} \sum_{i=1}^{N} s(i)$$

where N is the total number of data points.

22.4 Usage

The Silhouette Coefficient can be used to: - Determine the optimal number of clusters by comparing the average silhouette scores for different numbers of clusters. - Evaluate the quality of clustering algorithms, with higher scores indicating better-defined clusters.

22.5 Example

To compute the Silhouette Coefficient in Python, you can use the `silhouette_score` function from the `sklearn.metrics` module:

[ ]:
```python
from sklearn.metrics import silhouette_score

# X is your data and labels are the cluster labels
score = silhouette_score(X, labels)
print(f'Silhouette Score: {score}')
```

23 PCA: Principal Component Analysis in scikit-learn

- Reduces dimensions while trying to capture as much info from the dataset as possible

[ ]:
```python
from sklearn.decomposition import PCA

pca = PCA()
transformed = pca.fit_transform(X)
```

Transforms dataset along principal axes. The first axes tries to capture the maximum variance within the data. From here additional axes are constructed which are orthogonal to the previous axes and continue to account for as much of the remaining variance as possible.

Transforms this:

Into this:

[ ]:
```
pca.explained_variance_ratio_
```

```
array([9.99760273e-01, 2.39727247e-04])
```

Results are cumulative

[ ]:
```
np.cumsum(pca.explained_variance_ratio_)
```

```
array([0.99976027, 1.        ])
```

Below we visualize the first PCA component

23.1 Steps for Performing PCA

The theory behind PCA rests upon many foundational concepts of linear algebra. After all, PCA is re-encoding a dataset into an alternative basis (the axes). Here are the exact steps:

1. Recenter each feature of the dataset by subtracting that feature's mean from the feature vector
2. Calculate the covariance matrix for your centered dataset
3. Calculate the eigenvectors of the covariance matrix
    1. You'll further investigate the concept of eigenvectors in the upcoming lesson
4. Project the dataset into the new feature space: Multiply the eigenvectors by the mean-centered features

24 Market Segmentation with Clustering

- one of the most popular use cases for clustering

25 What is Market Segmentation?

- **Cluster Analysis** to segment a customer base into different *market segments* using the clustering techniques we've learned
- Ex: decide marketing budget allocation in order to attract more customers
    - Create personalized regression models for each group
- Know who your customer is. Identify sements in the customer data we can look for trends
    - Ex decide the station to run commercials on
- Find the segments with clustering
    - find them based on behavior

25.1 Targeting

Segmentation is just the first step

- Build individualized strategies
    - which market segment is most valuable to us? Use resarch and data analysis

  - how do we allocate the advertising budget? determine where to spend money best to reach the group
- Figure out how to position our product to make it both desirable and stand out from competitors

26 Natural Language Processing

26.1 Natural Language Tool Kit (NLTK)

Popular NLP library in Python

26.2 Regular Expressions

regex

26.3 Feature Engineering for Text Data

Text data has a lot of ambiguity and feature engineering for NLP is specific.
* How to remvoe stop words * create frequency distributions * representing histograms * stemming * lemmatization * bigrams which shows how often two words occur together

26.4 Context-Free Grammars and Part-of-Speech (POS) Tagging

- Context Free Grammar and Part of Speech tagging
- POS tagging helps a computer understand how to interpret a sentence
- Context free grammars (CFG) defines the rules of how sentences can exist.

26.5 Text Classification