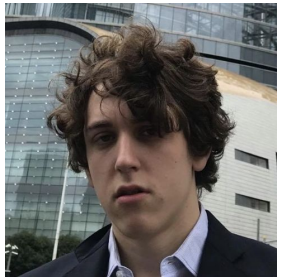
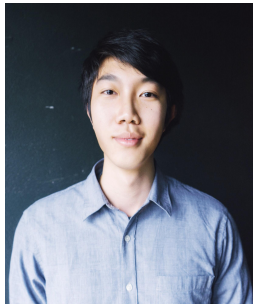
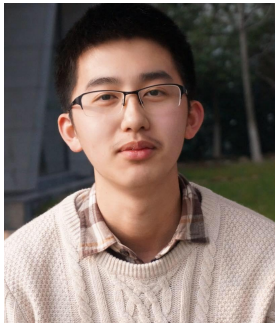
Project Marco

A Smarter Vacation

Team

 **Abhimanyu Choudhary** 2020	 **Ezra Max** 2020	 **Shiyu Tian** 2022	 **Hao Zhu** 2022

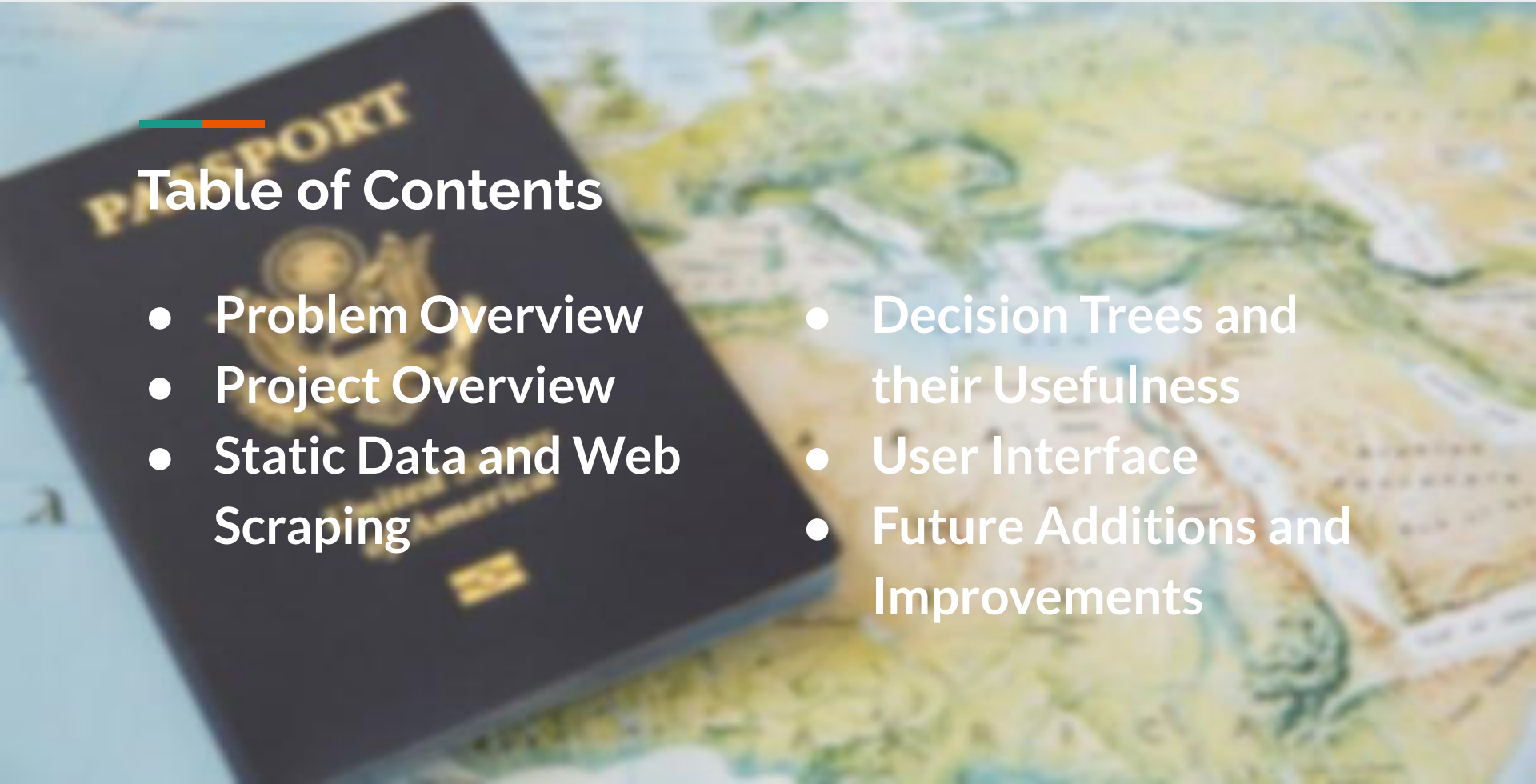


Table of Contents

- Problem Overview
- Project Overview
- Static Data and Web Scraping
- Decision Trees and their Usefulness
- User Interface
- Future Additions and Improvements

The Problem

How do you know where to go?

- It’s hard to gather the information you want about a vacation.
- You can’t know what you want from a vacation.
- How can we best present you one you want to buy?

Find Locations that fit user preferences for:

1. Weather and Climate
2. Activities and Attractions
3. Geographical Qualities
4. Safety and Language

While Scheduling:

1. Flights and Other Transport
2. Hotels and Accomodation
3. Activities the User would be interested in

Project Objective:

Given a user's answers to a few simple questions, can we suggest to them a vacation they will be interested in or intrigued by?

- Need to understand their preferences, preferably with minimum of input.
- Need to match vacation spots to those preferences.
- Need to present vacation data in an appealing, manageable way.

Can we surprise users with a fantastic vacation that they never would have considered?

Project Overview

Our Solution

How we generate a vacation:

- Activity info and binning to identify experiences
- Live weather, flight, hotel info + static language and safety info for logistics
- Assess this information through a few simple questions by using decision trees

How we present a vacation:

- UI, text, and visuals tailored to make process fun and results appealing

Code Overview

Data sources and APIs

- Descriptions and images
- Flights and flight pricing
- Activities and hotels
- Weather data
- Language data
- Travel Safety data
- Location data

Libraries and Packages

Selenium
Automated browsing for web scraping.

Beautiful Soup
Python web scraping for html only sites.

Pandas
Data storage and cleaning package.

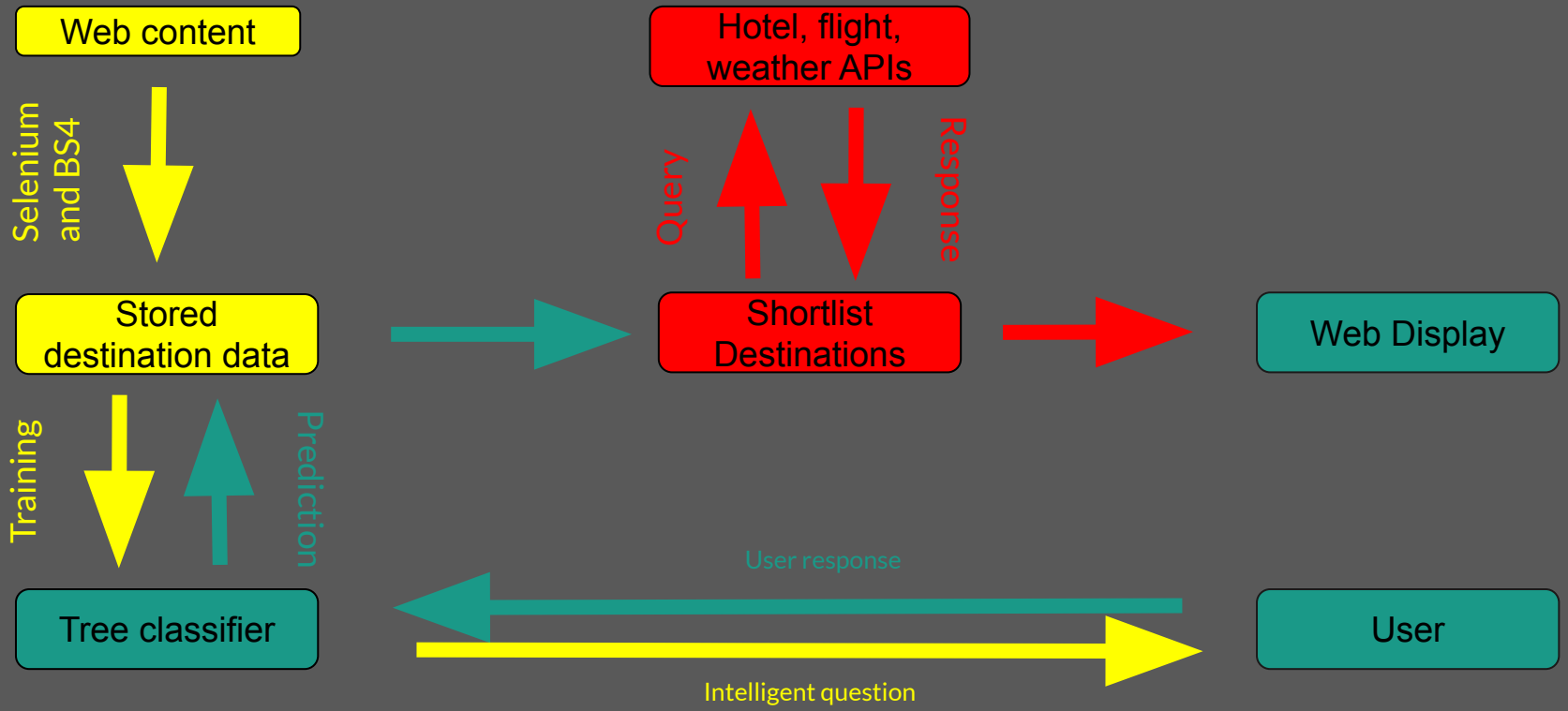
JSON
Data storage and exchange.

SciKit Learn
Decision Tree classification toolbox.

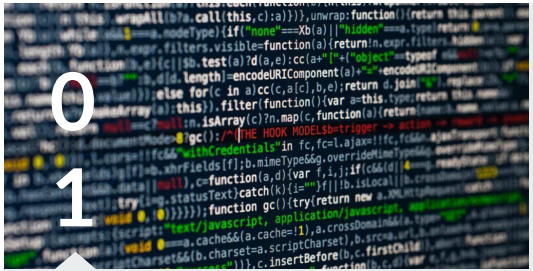
Requests
HTTP API requests package.

Pickle
Data serialization

Code Overview

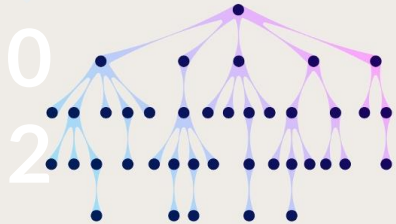


Our Development Process



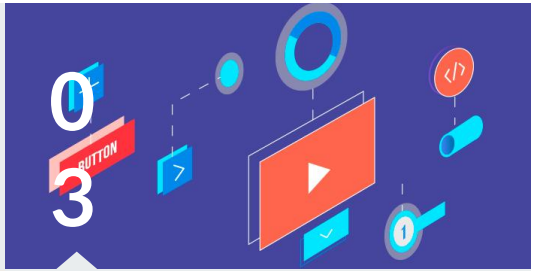
Gather and Scrape Static Data

Using a variety of APIs and web scraping utilities, we gathered various keywords and travel-related data associated to different cities around the world.



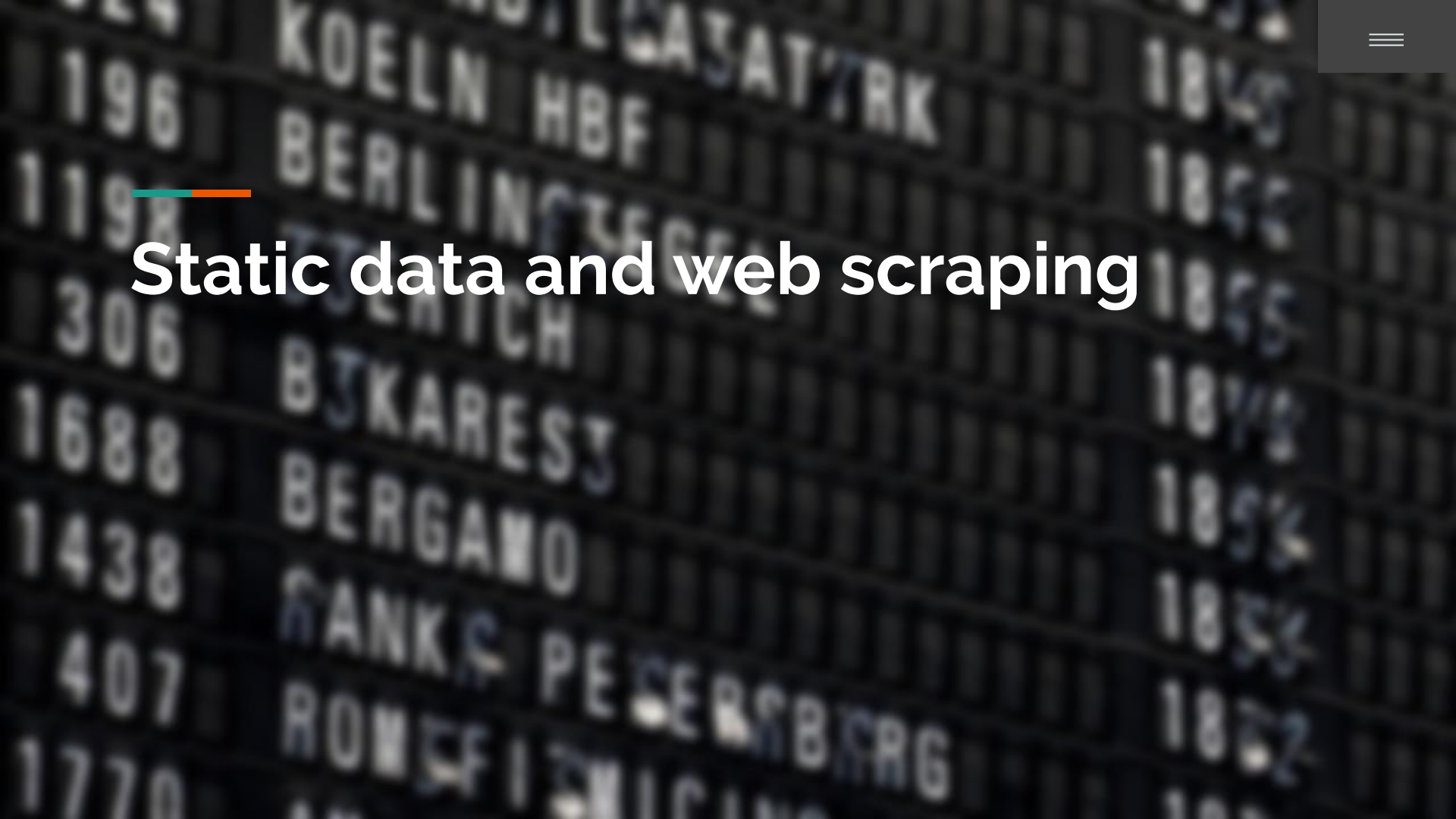
Build and Train a Decision Tree

Given all the data for each city, we computed a metric we call the “score” which quantifies how strong a city is in a certain regard. We then used these scores to train a decision tree. The output of this tree would be used to determine suitable vacation spots



Build a User Interface

Using Django Webserver, we integrated our decision tree along with our dynamic data sources, to provide one, cohesive piece of output.



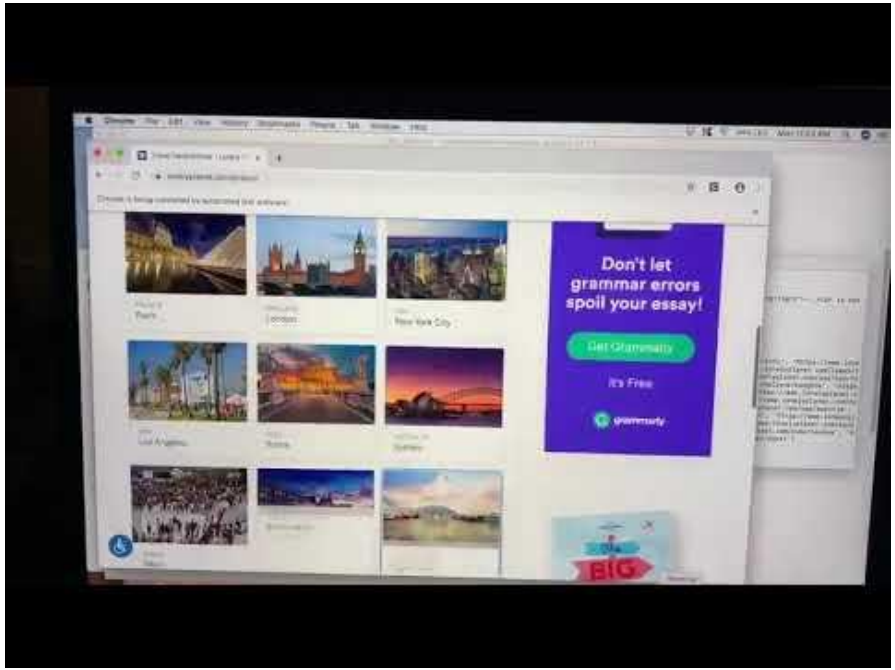
Static data and web scraping

Selenium and LonelyPlanet

- -We wanted to use LonelyPlanet for sentiment analysis via TextRazor (NLP API)
  - -This worked just OK, so we ended up stealing text and images from LonelyPlanet
  - -TextRazor didn't perform as well as we expected
- -Why selenium was necessary
  - -URLs are not consistent
  - -Clicks, windows to get there

Selenium and LonelyPlanet

- -At first, was working great!
- -Then, cookie warnings after too many queries. Learned Selenium the hard way getting around these.



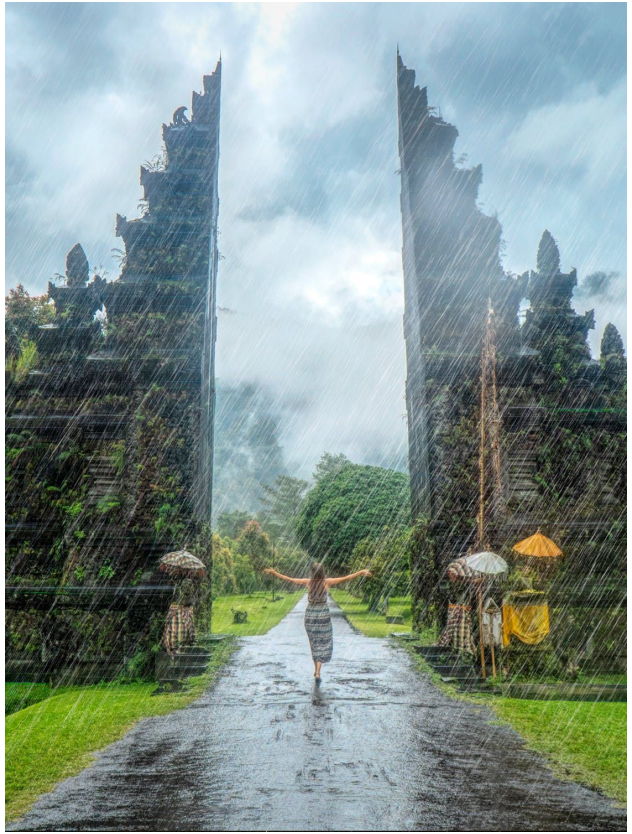
Weather APIs, flight APIs, Location APIs, TripAdvisor APIs

- -Used API to get activities in different categories and their counts from TripAdvisor. More reliable than textrazor approach.

- -Originally wanted to get flight and hotel info everywhere.
- -Main challenge query limits (shoulda known..)
- -Redesigned it and learned from failure (flights at end)
- -Needed departure and arrival info

Using BS4
Wikipedia,
State Department, etc.

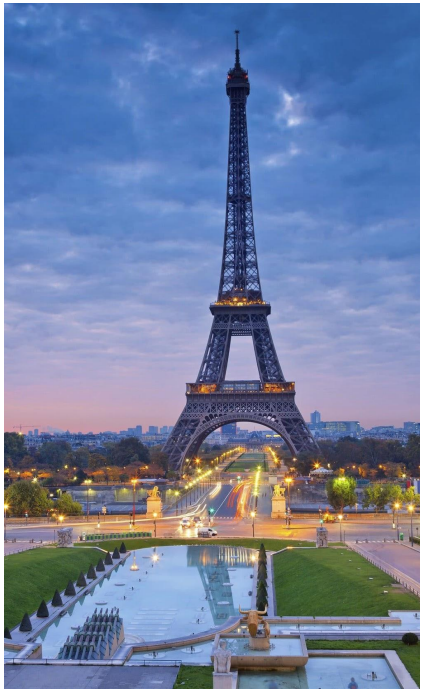
-Got language, safety information, etc. through BS4 scraping



Decision Trees

Vacations: A Classification Problem

- What is the best vacation for ***you***?
- More broadly:
  - Given a list of/profile of preferences, what is the best vacation destination?
    - Beach bums would certainly prefer Palma de Mallorca to Paris.
    - Architecture geeks might make the opposite decision.
    - The mountainous adventurer might prefer neither.



Paris or Palma?

Vacations: A Classification Problem

- Human psychology makes these decisions boundlessly complex and whimsical
  - Bob likes relaxing on the beach and drinking Piña coladas.
  - He does not like playing Beach volleyball or exploring coastal coves and reefs however.
  - How can we recommend Bob some vacations?

Bar or Barrier Reef?

What Questions should we ask you?

- We'd like to determine a client's favorite vacation, so we should "interview" them to glean some information regarding their preferences.
- Of course, we'd like our interview to be as succinct and accurate as possible.
  - We'd like to ask questions in an order so that each time we ask, we learn *as much as possible* with the client's response.
  - A bad conversation might be something like:
    - Do you like cold weather? - No
    - After asking - Do you like skiing? - Can you guess what this answer will be?

The ID3 and CART 4.5 Algorithms

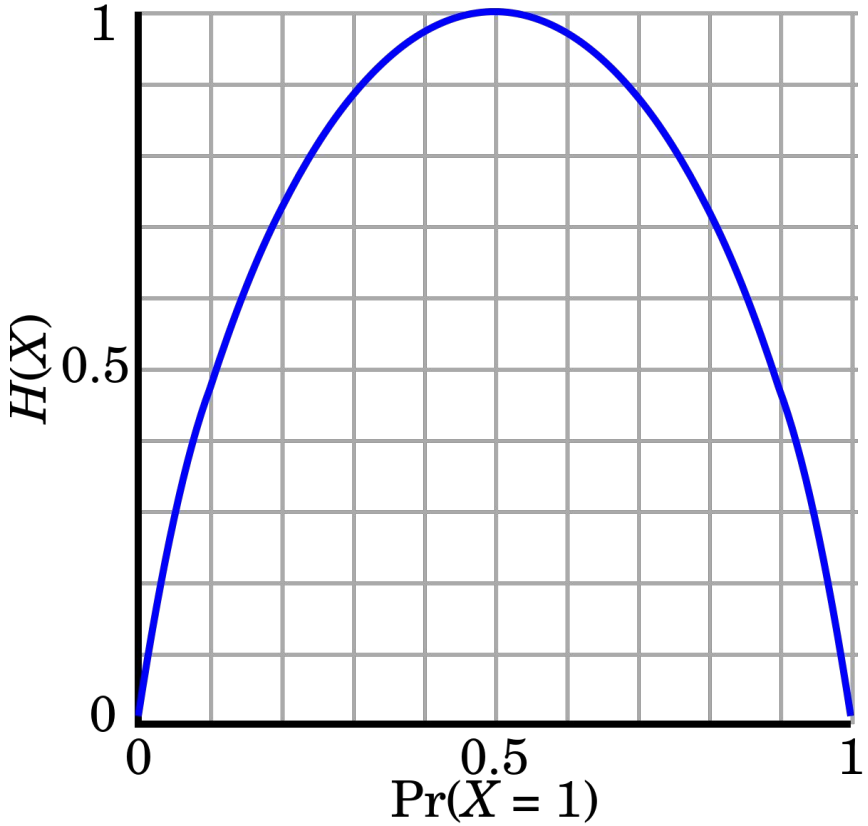
- Quantify the idea of the “next-best-question” through the information theoretic concept of Shannon Entropy.
- For a general discrete random variable, the definition is:

$$H(X) = \sum_{k=1}^{n} \mathbb{P}(X = x_i) \log \mathbb{P}(X = x_i)$$

- Reference: Shannon, Claude Elwood (July 1948). "A Mathematical Theory of Communication" (PDF). Bell System Technical Journal. 27 (3): 379–423
- This generalizes to sample data:
    - Replace the probabilities with sample proportions.

The ID3 and CART 4.5 Algorithms

- Intuitively: Measures the unpredictability of a state or its *average information content*.
  - For example, when political polls first occur, they provide us with lots of new information about voter preferences.
  - When we poll voters a second time after the first, *generally*, we obtain less information, i.e., the results become more predictable.
- In the context of ID3:
  - We want to find the attribute that nets us the most *information gain* in the sense of entropy maximization.
    - We can then **split** the tree on this attribute.
    - By splitting on this attribute, we gain the *most information* in terms what the user would be thinking of.



Sample Entropy as a function of p - The proportion of the data that can be classified as 1 in a binary (0,1) classification scheme. Maximized when the data is partitioned evenly, i.e p = 1/2 .

Scoring Locations

- We designed a system to score various locations based on keywords.
  - How “Historic” is Paris? How “good” are the Bahamas for boating?
- First Step - Collect TripAdvisor data and keywords regarding locations.
  - Collect Top 30 attractions for each city in our list.
  - Sorted by Trip Advisor ranking.
- Construct a categorization scheme for keywords.
  - Manually classified certain activities into customized buckets, based on desired granularity.
- Weighted each attraction location by its rank in the list.
  - If this was the number one ranked location, give it a full score of 1
  - Otherwise, (linearly) dock some points
  - Location 1 has score 1, location 30 has score 0.5

```
SEA_NATURE = ['Beaches', 'Bodies of Water', 'Reefs','Islands','Marinas','Swim with dolphins',\
'Duck tours','Bodies of Water','Caves','Dams','Waterfalls','Caverns/ Caves','Dams']


SEA_OUTDOOR = 'River rafting & tubing,Boat rentals,Dolphin & whale watching,\
Gondola cruises,Speed boats tours,Kayaking & canoeing,Stand-Up Paddleboarding,Shark diving,\
Scuba & snorkelling, Waterskiing & jetskiing,Swim with dolphins,Surfing,Windsurfing & Kitesurfing'.split(',')
```

A Sample Categorization

The ID3 and CART 4.5 Algorithms

- This graph makes sense: we “learn” the most about the data when observing it if it is evenly partitioned.
- This is how the ID3 (and CART 4.5) algorithm works.
    - Each time the tree splits, it does so on the attribute that partitions the dataset most evenly.
- We can understand our previous example in light of this framework.
    - If we know the user does not like cold weather, but we ask him about skiing, we will not partition the remainder of the dataset effectively.
    - Rather, we will leave it intact, because the tree will contain only warm weather activities.

SIGHTS_AND_LANDMARKS ≤ 5.918
entropy = 7.799
samples = 31220
value = [150, 134, 131, 139, 143, 150, 137, 148, 135, 131
139, 129, 150, 142, 141, 138, 145, 147, 136, 139
135, 137, 142, 146, 138, 138, 137, 140, 143, 141
136, 135, 134, 141, 134, 146, 152, 147, 141, 145
141, 147, 143, 145, 138, 146, 145, 125, 145, 145
145, 139, 138, 146, 137, 132, 138, 142, 130, 143
136, 138, 137, 143, 133, 148, 147, 156, 141, 148
140, 134, 142, 141, 143, 133, 141, 140, 142, 145
145, 148, 137, 142, 136, 143, 131, 147, 145, 148
142, 140, 143, 143, 144, 134, 133, 150, 145, 138
150, 142, 137, 137, 147, 129, 137, 141, 128, 135
134, 152, 131, 133, 133, 135, 144, 137, 142, 142
133, 138, 142, 137, 137, 138, 127, 139, 131, 150
125, 140, 140, 133, 140, 138, 136, 145, 150, 131
141, 138, 140, 150, 138, 138, 145, 147, 139, 141
138, 134, 132, 147, 132, 139, 146, 152, 139, 152
139, 143, 138, 148, 131, 141, 143, 149, 131, 139
149, 127, 139, 138, 138, 145, 133, 141, 132, 144
139, 129, 148, 147, 130, 131, 142, 142, 139, 134
140, 133, 139, 146, 148, 136, 135, 140, 143, 135
143, 141, 143, 141, 140, 148, 144, 142, 120, 132
129, 140, 139, 148, 129, 150, 150, 140, 148, 133
135, 150, 143]
class = oslo

True

False

NATURE_PARKS ≤ 6.203
entropy = 6.876
samples = 14763
value = [126, 126, 131, 139, 0, 0, 137, 117, 109, 0, 0
129, 0, 3, 0, 136, 145, 147, 0, 0, 38, 0, 0
138, 0, 0, 137, 140, 0, 2, 136, 0, 133, 0, 0
146, 144, 147, 0, 135, 141, 147, 143, 145, 138
143, 0, 125, 0, 0, 0, 122, 138, 146, 2, 4, 138
142, 0, 0, 136, 0, 0, 0, 133, 7, 140, 0, 10, 0
62, 0, 137, 141, 4, 14, 0, 0, 142, 15, 145, 1
0, 9, 98, 0, 131, 0, 145, 0, 142, 140, 143, 132
0, 56, 133, 150, 0, 134, 146, 142, 0, 137, 0
122, 137, 139, 49, 31, 134, 0, 12, 0, 133, 135
144, 0, 16, 0, 0, 0, 105, 0, 137, 0, 6, 139
110, 150, 24, 138, 0, 133, 131, 138, 0, 25, 1
131, 141, 0, 0, 150, 0, 0, 32, 19, 12, 0, 0, 0
88, 147, 0, 40, 1, 2, 139, 152, 15, 0, 0, 0
131, 138, 91, 0, 48, 139, 0, 127, 0, 1, 138, 141
1, 141, 0, 7, 12, 37, 0, 97, 130, 0, 0, 0, 139
0, 0, 133, 0, 0, 148, 136, 129, 0, 19, 51, 140
0, 142, 0, 2, 0, 46, 142, 120, 0, 129, 0, 0
148, 123, 150, 0, 140, 148, 133, 79, 150, 139]
class = vienna

SIGHTS_AND_LANDMARKS ≤ 9.748
entropy = 7.005
samples = 16457
value = [24, 8, 0, 0, 143, 150, 0, 31, 26, 131, 139, 0
150, 139, 141, 2, 0, 0, 136, 139, 97, 137, 142
8, 138, 138, 0, 0, 143, 139, 0, 135, 1, 141, 134
0, 8, 0, 141, 10, 0, 0, 0, 0, 0, 3, 145, 0
145, 145, 145, 17, 0, 0, 135, 128, 0, 0, 130
143, 0, 138, 137, 143, 0, 141, 7, 156, 131, 148
78, 134, 5, 0, 139, 119, 141, 140, 0, 130, 0
147, 137, 133, 38, 143, 0, 147, 0, 148, 0, 0, 0
11, 144, 78, 0, 0, 145, 4, 4, 0, 137, 0, 147
7, 0, 2, 79, 104, 0, 152, 119, 133, 0, 0, 0
137, 126, 142, 133, 138, 37, 137, 0, 138, 121, 0
21, 0, 101, 2, 140, 0, 9, 0, 136, 120, 149, 0
0, 138, 140, 0, 138, 138, 113, 128, 127, 141, 138
134, 44, 0, 132, 99, 145, 150, 0, 0, 124, 143
138, 148, 0, 3, 52, 149, 83, 0, 149, 0, 139, 137
0, 4, 132, 0, 132, 137, 127, 92, 148, 50, 0, 131
142, 142, 0, 134, 140, 0, 139, 146, 0, 0, 6, 140
124, 84, 3, 141, 1, 141, 138, 148, 98, 0, 0, 132
0, 140, 139, 0, 6, 0, 150, 0, 0, 0, 56, 0, 4]
class = oslo

A (fairly shallow) example decision path

Statistical Caveats and Assumptions

- Any use of machine learning involves implicit assumptions on data to be effective.
- In particular, any model needs to be *trained* properly in order to have any sort of prediction or classification power.
- In our case: we would need users to prescribe scores and pick cities **ahead of time**.
    - This way, the model already has some semblance of the classification structure.

Statistical Caveats and Assumptions

- In our case, it would have be infeasible to obtain proper training data in time.
- Thus, we used Monte-Carlo sampling to generate a sufficient amount of training and testing data.
- In particular, we added normally distributed perturbations to scores.
    - This assumes that scores are normally distributed.
- This assumption can be inaccurate for a variety of reasons.
- For example, scores could have too much “tail-heaviness”
    - People only put ratings if their experience is really good or bad.

Decision Tree Specific Issues

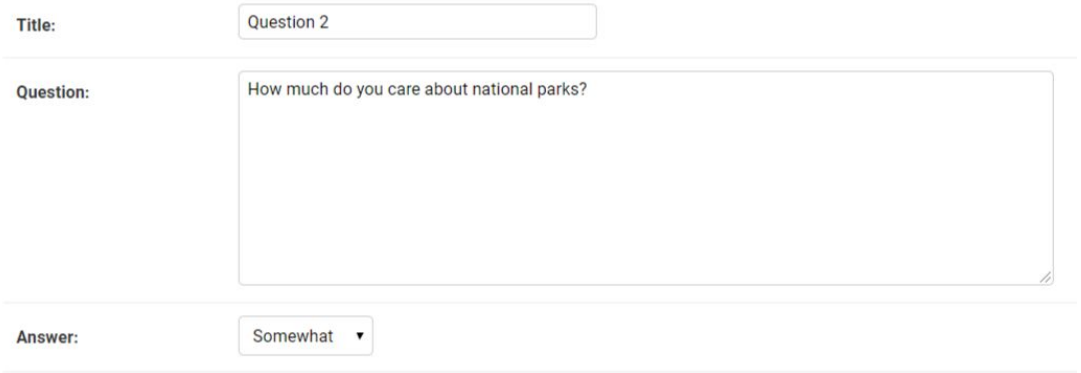
- Decision Trees fall on one end of the Bias-Variance tradeoff.
  - They are *very* high variance objects, with low bias.
  - This means they are prone to overfitting on training data. In our case, the tripadvisor and keyword data.
    - Small perturbations of initial data can change the model entirely.
- They also suffer from multicollinearity.
  - When two related variables (for example, weather and beach activities) might explain the same response, the tree will greedily pick the one that maximizes entropy.
  - This of course might not be realistic.
- We aimed to address this in part by limiting the depth of our model: 10 levels deep at most.
- We will discuss our future plans to avoid such issues later in the presentation.

User Interface

Django

Questions:

- Regular Question
- Background Information Question



Forms

- Questions Form
- Forms related to background information

Render & HTML

Pass in variables generated by functions to display in urls

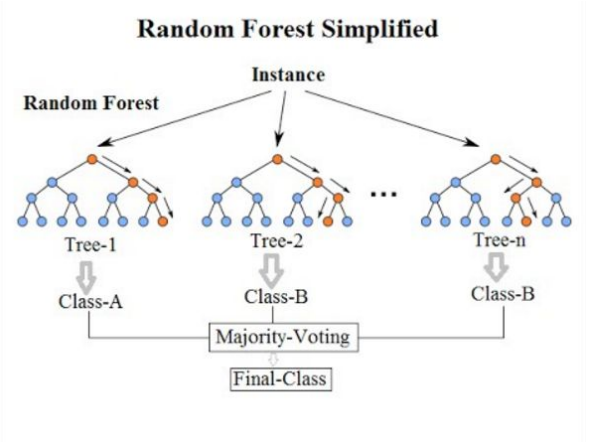
Demonstration

Future Additions and Improvements

Statistical Improvements

- Ideally, train the dataset using real, human data.
- Avoid high variance/overfitting of decision trees.
  - Exchange some variance for bias.
  - Use bagged trees or possibly random forests.

Statistical Improvements



- Random Forest
  - Collection of Decision Trees
  - Each built by training decision trees on subsets of the data.
    - Each using different features, training data, etc.
    - Each tree then votes/posits a class.
    - Winner is the majority vote.
  - Would be useful for granularity concerns.
  - Higher Bias, lower variance.

Statistical Improvements

- Further research on properly scoring vacations.
  - What is a good metric?
  - How to decay based on number of other attractions in city?
  - How to take into account the diversity of reviews?

Data Diversification

- Expand corpus for classification of vacations.
  - Currently only using TripAdvisor and Lonely Planet Datasets.
  - Trip Advisor and its reviews are the only source for attraction/classification info.
  - Find different NLP to better match activities and specific attractions.
    - Possible Tools: ConceptNet, WordNet.
- Expand sources for travel and hotel information.
  - Current API choices limited to TripAdvisor and Kiwi.com
  - Find more hotel, attraction, and flight information.
- Dynamification
  - Make things as dynamic as possible, with some caching to speed things up.

Interface Improvements

- Add more goodies to the interface
  - Dynamic Weather Data and Pictures of Weather
  - Hotel booking links, pictures, and reviews.
    - Currently hotels are picked by price, maybe pick by reviews + price
  - Add different attractions in cities based on user input and scores.

Thank you.