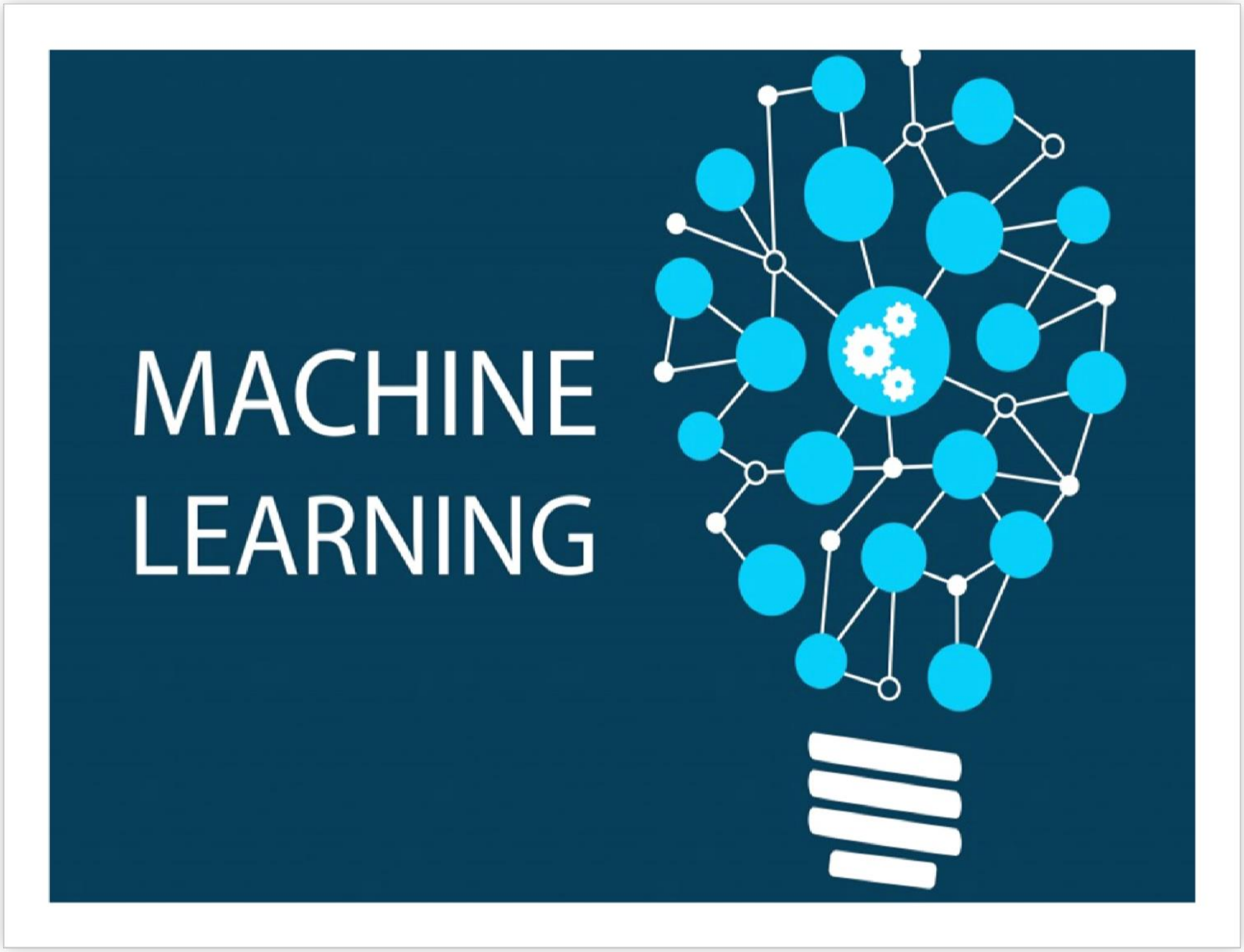


Pokémon Go

Submitted to: Yogendra Sir
Submitted by: Shronit Bhargava & Sachin Sharma

TABLE OF CONTENT

- Certificate
- Acknowledgement
- Abstract
- List of figures
- Introduction
- Theory
- Methodology
- Data Pre-processing
- Regression Method Used
- Classification Method used
- Data visualization
- Result Analysis
- Annexure

Certificate

Date: 27/06/17

This is to certify that Mr. **Shronit Bhargava**, student of 3rd Year from Department of Computer Science, NIIT University, Neemrana, has undergone a Project work from May 29, 2017 to July 05, 2017 in **Data Science & Machine Learning** titled "**Pokémon Go: Gotta Predict'em All**"

________________________ ________________________

Project Incharge Seal

Certificate

Date: 27/06/17

This is to certify that Mr. **Sachin Sharma**, student of 3rd Year from Department of Computer Science, SVKM's NMIMS Mukesh Patel School of Technology Management and Engineering has undergone a Project work from May 29, 2017 to July 05, 2017 in **Data Science & Machine Learning** titled "**Pokémon Go: Gotta Predict'em All**"

__________________ __________________

Project Incharge Seal

Acknowledgement

For making this project we would like to thank our course instructor, Yogendra Sir, he has helped us in learning the various concepts of the Machine Learning. Here we would also like to thank Kunal Sir and Rohit Sir for mentoring us throughout this project.

Abstract

Rapid growth of data comes with a challenge of sorting and analyzing them, where raw data exists in graphical form, textual form or in images. Data science and machine learning finds its application in various fields like stock market, recommendation systems, image processing, aerial photography, military, weather forecasting etc.

This report is about our project on “Predicting Pokemon” that addresses about data pre-processing and post processing which includes plotting, classification and prediction of pokemon appearance in real time and the ability of machine learning algorithms to deal with different set of data. In this project, we have tackled a classification problem of predicting where the pokemon will appear by accessing several data variable like latitude, longitude, etc. We have tested and used Logistic Regression, Decision Tree, k-Nearest Neighbor and random forest to determine the results. In addition to this, we have also made use of several libraries to plot the data points on the map.

List Of Figures

Fig. 1 Data Science

Fig. 2 Decision Tree

Fig. 3 Logistic Regression

Fig. 4 Linear Regression

Fig. 5 Pokemon Activity

Fig. 6 Pokemon Activity Count in Various Cities

Fig. 7 Mapping of Pokemon Names with the ID’s

Introduction

Pokémon Go, a location-based, augmented reality mobile game released by Niantic Inc. The players use the GPS to locate, capture, and battle fictional creatures in a virtual setting who appear on the screen as if they were in the same real-world location as the player. Like many millennials, I grew up watching Pokémon. So, when this game was released, I was undoubtedly excited, running on the streets of Jaipur looking for the rarest Pokémon's (but always ended up with common Pokémon's) Although the game was a huge success, there were many flaws that needed to be addressed which ultimately caused the downfall of it. The target is to train a machine-learning algorithm so that it can predict where pokemon will appear in future.

Our approach to the problem is very simple, we first retrieve the dataset that consists of roughly 293,000 pokemon sightings (historical appearances of Pokemon) having coordinates, time, weather, population density, distance to pokestops, gyms etc. Then we will check the dataset for some missing data i.e. the data pre-processing part. Then, we move towards data fetching followed by importing features, splitting dataset, applying machine-learning algorithms, plotting the data points on map, classification and finally prediction.

Finally, we compare the performance of each of these algorithms with the help of accuracy matrices, score matrices and then chooses the one that gives more accuracy.

Theory

Data science is a "concept to unify statistics, data analysis and their related methods" in order to "understand and analyze actual phenomena" with data.[3] It employs techniques and theories drawn from many fields within the broad areas of mathematics, statistics, information science, and computer science, in particular from the subdomains of machine learning, classification, cluster analysis, data mining, databases, and visualization.

Data science – discovery of data insight

This aspect of data science is all about uncovering findings from data. Diving in at a granular level to mine and understand complex behaviors, trends, and inferences. It's about surfacing hidden insight that can help enable companies to make smarter business decisions.

For example:

Netflix data mines movie-viewing patterns to understand what drives user interest, and uses that to make decisions on which Netflix original series to produce

Data science – development of data product

A "data product" is a technical asset that: (1) utilizes data as input, and (2) processes that data to return algorithmically generated results. The classic example of a data product is a recommendation engine, which ingests user data, and makes personalized recommendations based on that data.

For example:

Amazon's recommendation engines suggest items for you to buy, determined by their algorithms. Netflix recommends movies to you. Spotify recommends music to you.

Machine learning and statistics are part of data science. The word learning in machine learning means that the algorithms depend on some data, used as a training set, to fine-tune some model or algorithm parameters. This encompasses many techniques such as regression, naive Bayes or supervised clustering.

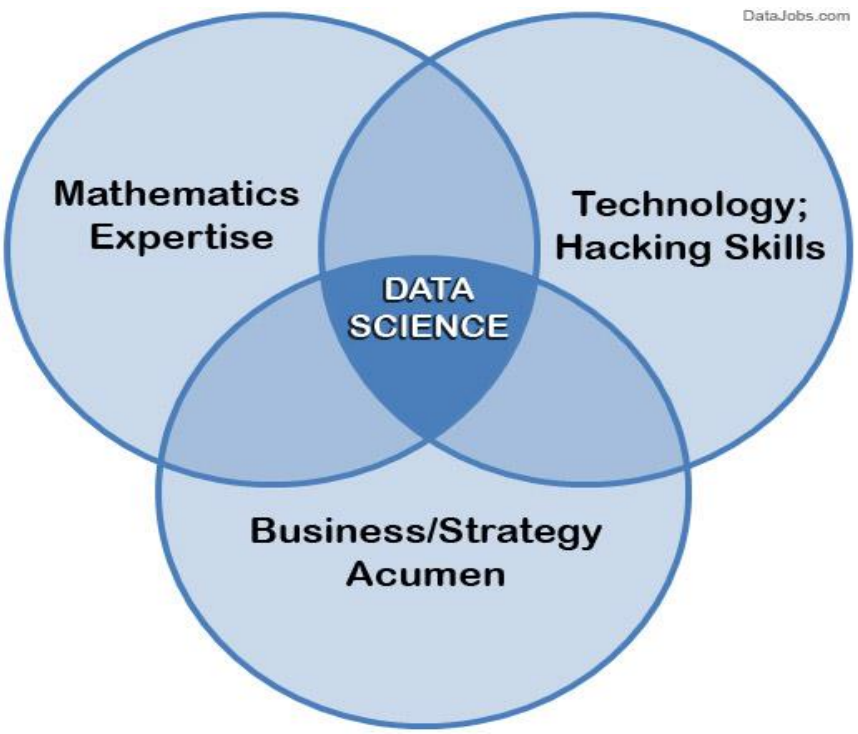


Fig. 1 Data Science

Supervised and unsupervised learning describe two ways in which machines algorithms can be set loose on a data set and expected to learn something useful from it.

Supervised:

If we are training our machine-learning task for every input with corresponding target, it is called supervised learning, which will be able to provide target for any new input after sufficient training. Our learning algorithm seeks a function from inputs to the respective targets. If the targets are expressed in some classes, it is called *classification* problem. Alternatively, if the target space is continuous, it is called *regression* problem.

- **Regression** analysis is widely used for prediction and forecasting, where its use has substantial overlap with the field of machine learning. Regression analysis is also used to understand which among the independent variables are related to the dependent variable, and to explore the forms of these relationships. In restricted circumstances, regression analysis can be used to infer causal relationships between the independent and dependent variables.

- **Classification** model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. Outcomes are labels that can be applied to a dataset.

Unsupervised: If we are training our machine-learning task only with a set of inputs, it is called unsupervised learning, which will be able to find the structure or relationships between different inputs. Most important unsupervised learning is *clustering*, which will create different cluster of inputs and will be able to put any new input in appropriate cluster.

- **Cluster** analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics.

1. **Decision Trees:** A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance-event outcomes, resource costs, and utility.

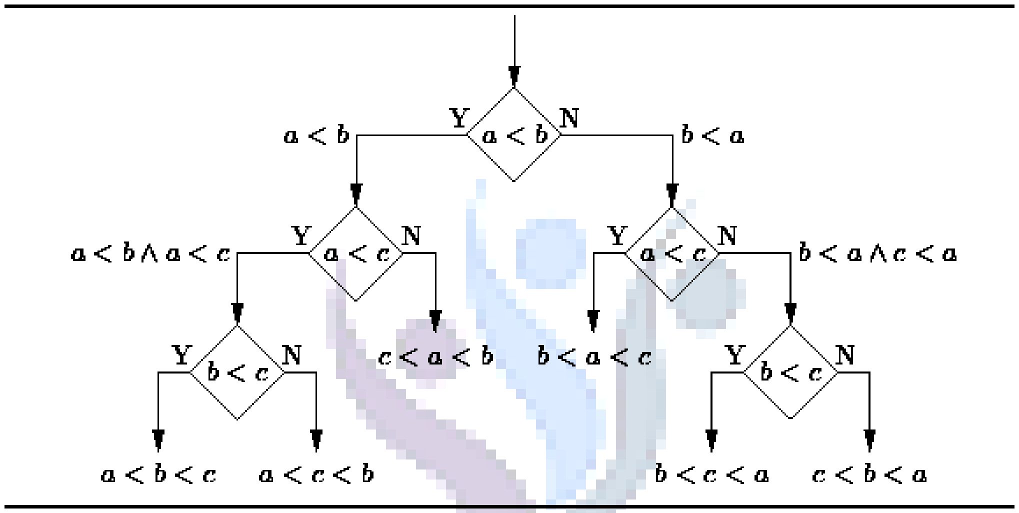


Fig. 2 Decision Tree

From a business decision point of view, a decision tree is the minimum number of yes/no questions that one has to ask, to assess the probability of making a correct decision, most of the time. As a method, it allows you to approach the problem in a structured and systematic way to arrive at a logical conclusion.

2. **Logistic Regression:** Logistic regression is a powerful statistical way of modeling a binomial outcome with one or more explanatory variables. It measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative logistic distribution.

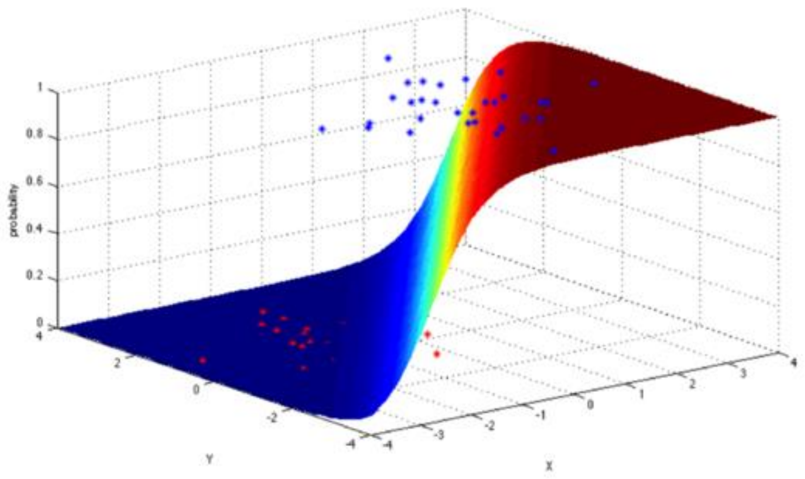


Fig. 3 Logistic Regression

3. Linear Regression

It is used to estimate real values (cost of houses, number of calls, total sales etc.) based on continuous variable(s). Here, we establish relationship between independent and dependent variables by fitting a best line. This best fit line is known as regression line and represented by a linear equation Y= a *X + b.

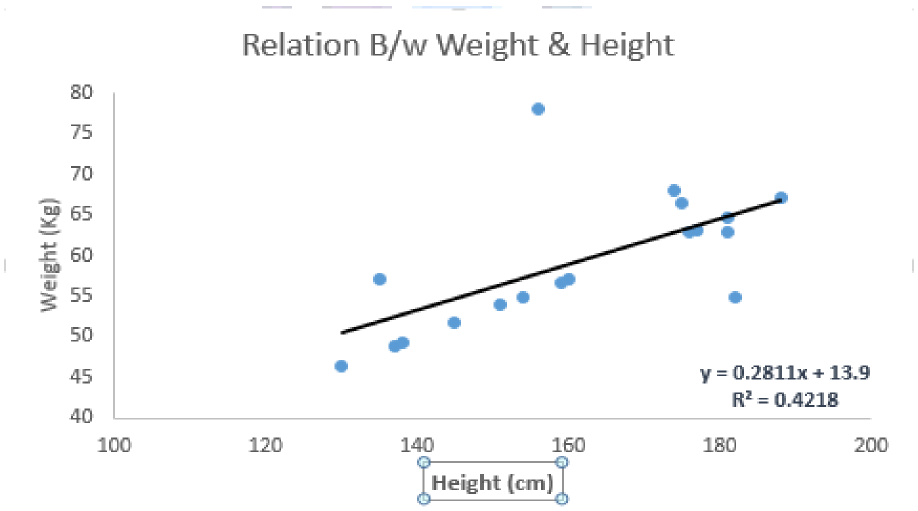


Fig. Linear Regression

4. KNN (K- Nearest Neighbors)

It is also a lazy algorithm. What this means is that it does not use the training data points to do any generalization. K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases by a majority vote of its k neighbors. The case being assigned to the class is most common amongst its K nearest neighbors measured by a distance function. These distance functions can be Euclidean, Manhattan, Minkowski and Hamming distance. First three functions are used for continuous function and fourth one (Hamming) for categorical variables. If K = 1, then the case is simply assigned to the class of its nearest neighbor. At times, choosing K turns out to be a challenge while performing KNN modeling.

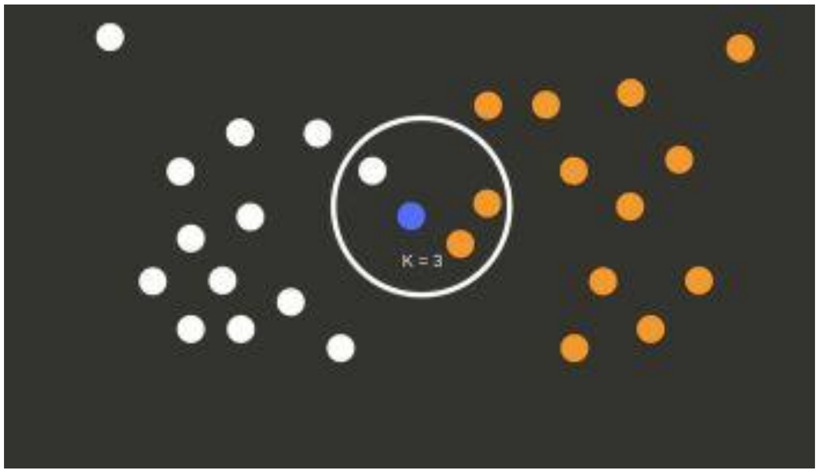


Fig. 4 KNN

Methodology

To understand the methodology adopted, we first understand our dataset and the variables.

Dataset

Our dataset consists of roughly 293,000 pokemon sightings (historical appearances of Pokemon), having coordinates, time, weather, population density, distance to pokestops/ gyms etc. as features. The target is to train a machine-learning algorithm so that it can predict where pokemon appear in future.

Features:

• pokemonId - the identifier of a pokemon, should be deleted to not affect predictions. (numeric; ranges between 1 and 151).

• latitude, longitude - coordinates of a sighting (numeric)

• appearedLocalTime - exact time of a sighting in format yyyy-mm-dd'T'hh-mmss.ms'Z' (nominal)

• cellId 90-5850m - geographic position projected on a S2 Cell, with cell sizes ranging from 90 to 5850m (numeric)

• appearedTimeOfDay - time of the day of a sighting (night, evening, afternoon, morning)

• appearedHour/appearedMinute - local hour/minute of a sighting (numeric)

• appearedDayOfWeek - week day of a sighting (Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday)

• appearedDay/appearedMonth/appearedYear - day/month/year of a sighting (numeric)

• terrainType - terrain where pokemon appeared described with help of GLCF Modis Land Cover (numeric)

• closeToWater - did pokemon appear close (100m or less) to water (Boolean, same source as above)

• city - the city of a sighting (nominal) • continent (not always parsed right) - the continent of a sighting (nominal)

• weather - weather type during a sighting

• temperature - temperature in celsius at the location of a sighting (numeric)

• windSpeed - speed of the wind in km/h at the location of a sighting (numeric)

• windBearing - wind direction (numeric) • pressure - atmospheric pressure in bar at the location of a sighting (numeric)

• weatherIcon - a compact representation of the weather at the location of a sighting (fog, clear-night, partly-cloudy-night, partly-cloudy-day, cloudy, clear-day, rain, wind)

• sunriseMinutesMidnight-sunsetMinutesBefore - time of appearance relatively to sunrise/sunset Source

• population density - what is the population density per square km of a sighting (numeric, Source)

• urban-rural - how urban is location where pokemon appeared (Boolean, built on Population density, <200 for rural, >=200 and <400 for midUrban, >=400 and <800 for subUrban, >800 for urban)

• gymDistanceKm, pokestopDistanceKm - how far is the nearest gym/pokestop in km from a sighting? (numeric, extracted from this dataset)

• gymIn100m-pokestopIn5000m - is there a gym/pokestop in 100/200/etc meters? (Boolean)

• cooc 1-cooc 151 - co-occurrence with any other pokemon (pokemon ids range between 1 and 151) within 100m distance and within the last 24 hours (Boolean)

• class - says which pokemonId it is, to be predicted.

Dependent Variables: latitude, longitude, appearedLocalTime, closeToWater, city, continent, weather, population, windspeed, closetowater, urban-rural, terraintype.

Independent Variables: pokemonid, class

Data Pre-processing

For pre-processing, we considered Python as our options for the project. After some experimentation, we found that while R was easier for statistics and analysis of the data, the lack of uniformity among the various ML packages made Python our preference. The ML algorithms provided by the scikit-learn package do not function if the input data has missing values. Hence we either had to impute data at the missing slots or remove the instances that had these missing fields. Upon examining the data, we realized that in this project we don't have any missing data but have categorical data that needs to be converted in integer values to make the model work. And to do that we used pd.get_dummie() and LabelEncoder().

Additionally, the statistical correlation between some features (for instances where they were available) were appeared to be not related. Hence, we dropped those features. In addition to this, sampling was necessary for machine learning algorithms.

Libraries used

- numpy
- pandas
- matplotlib.pyplot
- mpl_toolkits.basemap
- geopy.geocoders
- seaborn
- math
- pylab
- pickle
- scikitlearn
  - sklearn.cross_validation
  - sklearn.preprocessing

- sklearn.linear_model
- sklearn.neighbors
- sklearn.ensemble
- sklearn.metrics

Regression Method Used

Here in this project we had used Logistic Regression. Here we had used Logistic Regression 2 times, first to train the data for Latitude part and another time for the Longitude part. However as the data is very large and we had many attributes so the training part of the data took very long time. So we had trained the data single time only and had saved the trained data in the pickle file, from where we can use that train data to make the predictions. Here we had used Logistic Regression only because we were very much familiar with this method, so it was easy for us to use it and our requirements of doing predictions completely met with the features of the Logistic Regression.

Classification Method used

Here for making classification of the pokemons on the basis latitudes and longitudes we had used various algorithms like K – Nearest Neighbors, Logistic Regression in Classification mode, Decision Tree Classifier, Random Forest Classifier, SVM, but we got the highest accuracy that of 19.65% from the K - Nearest Neighbors, that too with 40 nearest neighbors.

Data visualization

For doing visualization of the data we had used libraries like Basemap, Seaborn and matplotlib. Through basemap we were able to easily map the different countries and continents and then on the basis of the longitude and latitude we were able to plot the pokemon through scatter plots. And for making the plot for counting the number of pokemons county wise we had used seaborn library and with its help we had use matplotlib to make the bar chart.

Result Analysis

Part 1

Here we had to plot the pokemons based on latitudes and longitudes.

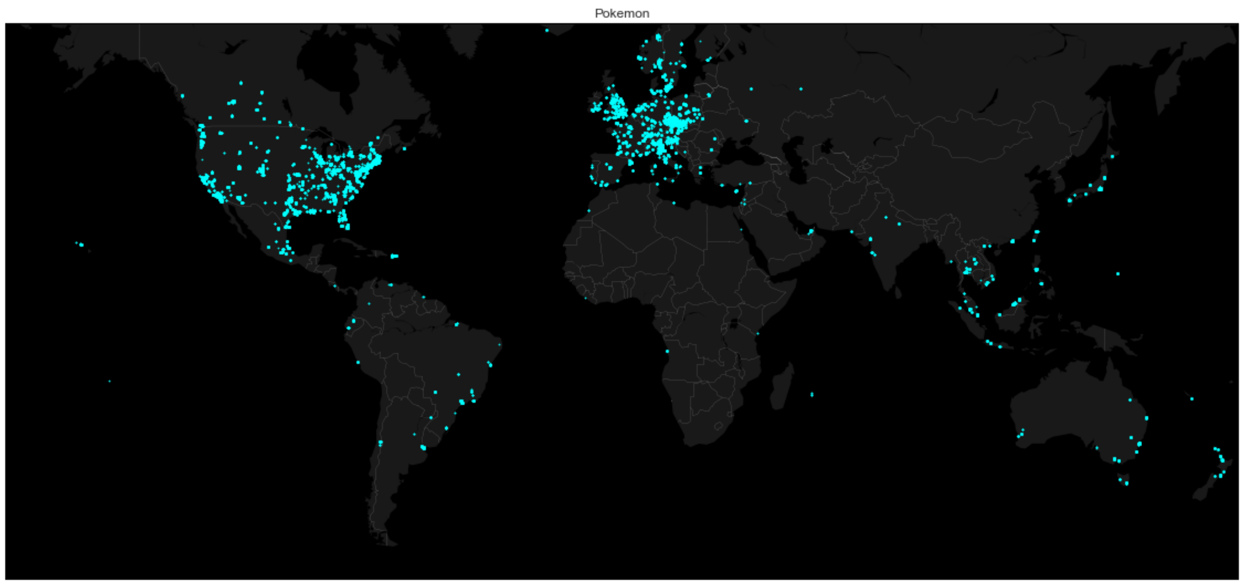


Fig. 5 Pokemon activity

Here red dots represent the pokemons present.

Part 2

In this part we had to make a bar chart of the number of pokemons present in the different cities.

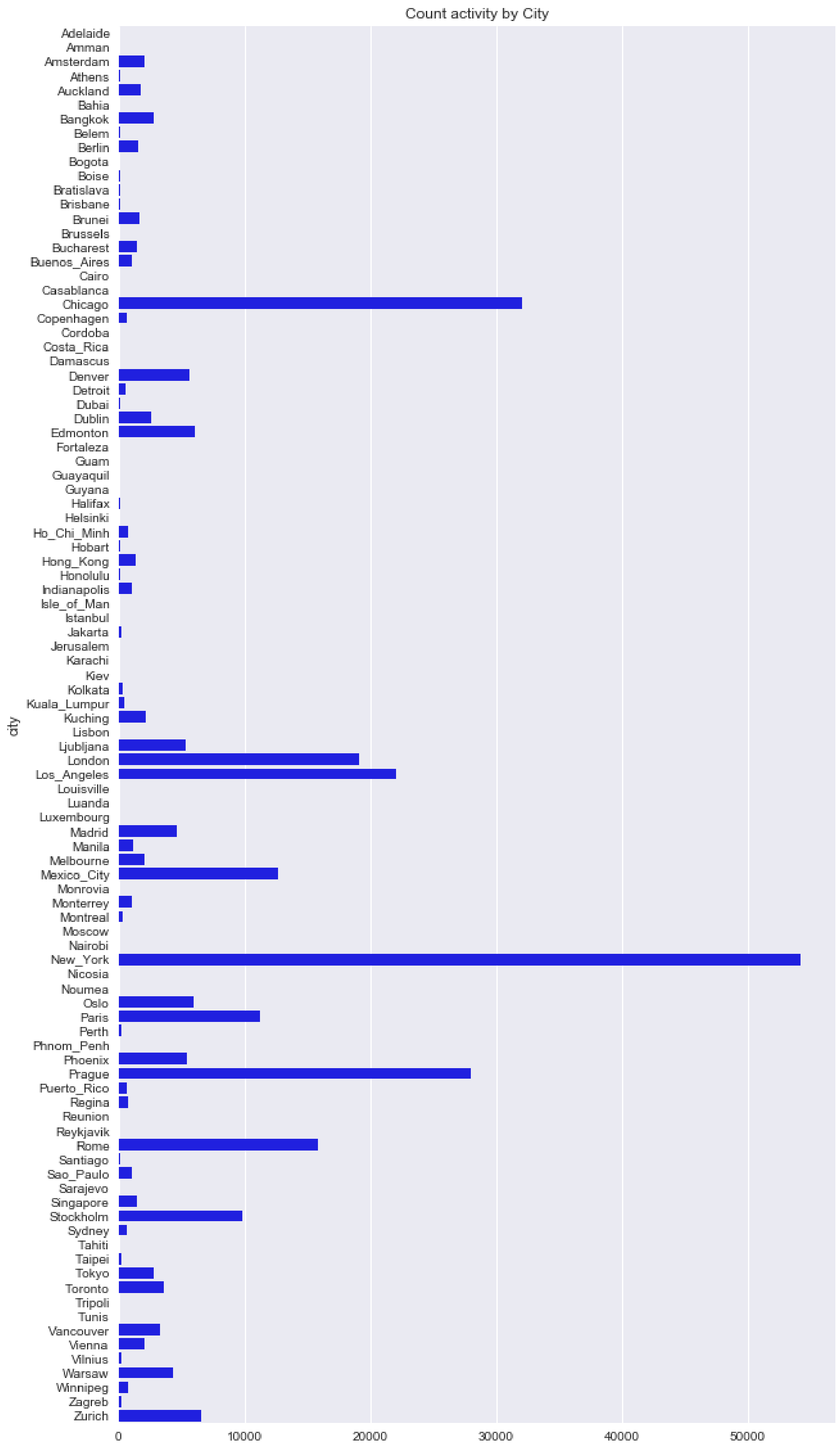


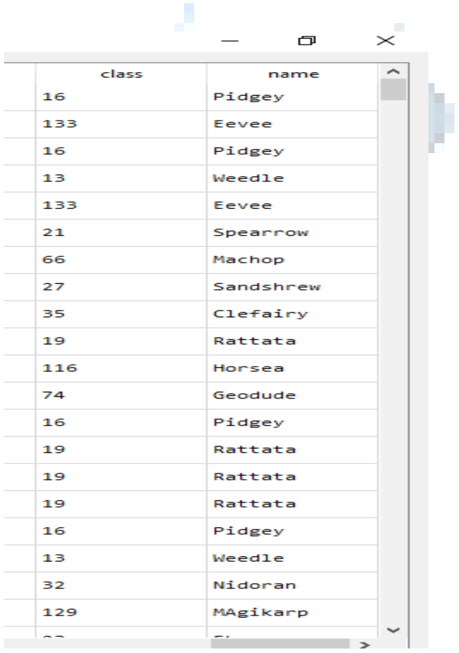
Fig. 6 Pokemon activity count in various citites

Part 3

In this part we had to make the classification of the pokemons based on latitudes and longitudes, so here we had used K – Nearest neighbors to classify them with 40 nearest neighbors, but here we got the best accuracy score of 19.65% only as the number of features used to classify the data was very less, resulting in low accuracy score.

Part 4

In this part we had to map the names of the pokemons with their ID's. So to do this task we had used the map function to map names with the ID's



class	name
16	Pidgey
133	Eevee
16	Pidgey
13	Weedle
133	Eevee
21	Spearrow
66	Machop
27	Sandshrew
35	Clefairy
19	Rattata
116	Horsea
74	Geodude
16	Pidgey
19	Rattata
19	Rattata
19	Rattata
16	Pidgey
13	Weedle
32	Nidoran
129	MAgikarp

Fig. 7 Mapping of names with ID

Part 5

In this part we had to predict the location of the pokemon of our choice. So for this purpose we had used Logistic Regression, here we got the accuracy score of around 74.5 % in total. But training time was very high but when using the trained data from the pickle file the computation time for giving the results was less than 1 minute.

Conclusion

Here in this project we were able to correctly visualize the pokemons found at various places around the world and also their activities in various cities. We were able to correctly classify the pokemons based on latitudes and longitudes although accuracy score for classification was not that good, but good enough with the amount of data present with us and the features used to make classification. We were able to correctly map the names of the pokemons with their ID’s. We were also able to make the predictions of the location of a particular pokemon with the accuracy of about 72.5%.

Annexure

Code for visualization :-

```
import pandas as pd

import matplotlib.pyplot as plt

from mpl_toolkits.basemap import Basemap


plt.figure(1, figsize=(20,10))

m1 = Basemap(projection='merc', llcrnrlat=-60, urcrnrlat=65, llcrnrlon=-180, urcrnrlon=180, lat_ts=0,
resolution='c')


m1.fillcontinents(color='#cc9966',lake_color='#99ffff')

m1.drawmapboundary(fill_color='#99ffff')

m1.drawcountries(linewidth=0.1, color="w")


x, y = m1(df.longitude.tolist(),df.latitude.tolist())

m1.scatter(x,y, s=3, c="red", lw=0, zorder=5)
```

Code for training the data with Logistic Regression

```
from sklearn.linear_model import LogisticRegression

clas = LogisticRegression(random_state = 0)

clas1 = LogisticRegression(random_state = 0)

print 'Training data with latitudes...'

clas.fit(X_train, Y_train)

print 'Training data with longitudes...'
```

```
clas1.fit(X_train, Y1_train)
```