CSC311 Project Report

Kaifeng Li, Kangzhi Gao, Keke Chen, Xiaoyi Jia

April 8, 2024

1 Introduction

2 Data Exploration

Before we perform any pre-process strategies, we will first take a look at the distribution of each given feature in the dataset.

The original dataset contains 10 features. based on the questionnaire, the first four

Based on the original dataset, we need to do some cleaning processes to improve the prediction performance, or even be able to use the model. Also, real-world data is very likely to have outliers and missing values. Because our sample size is small, we decided to handle all the missing values by filling them with reasonable values.

For questions 1-5, because these features in the dataset only have 5 or 4 different inputs, we use Onehot encoding to transform them into extra features. If there are missing values exist in these categorical features, the missing values of questions 1-4 will be filled by 2, and the missing values of question 5 will be filled as one of the possible choices. We don't randomly choose a value for the missing value of the first 4 questions because the original setting of these features has a meaningful numerical size, and 2 is closer to the "mean". For questions 7-9, we will treat them as numerical explanatory variables. Based on this semester's lab, we know that normalization will improve the prediction accuracy. So, we fill all the missing values by the mean of each feature and normalize them. For question 6, the question asked participants to rank the given 6 keywords by correlation with the city. However, some participants are not ranking them, they rating these keywords. So, in some samples, the values are repeated. So, we have three different strategies for handling this misunderstanding. First, we only take the keyword with the maximum score and the keyword with the minimum score as input to the model. If two or more keywords have the same value, we randomly choose one of them. Second, we treat this feature as 6 numerical variables. We create 6 new variables and each variable represents one of the keywords, the value of each new variable will be the corresponding score in the original feature. The two strategies listed above will share the same missing value handling process, the missing values will be filled by 3.5 (mean), because they all treat the value of the original feature as numerical. Third, we ignore the numerical size for the actual ranking (a bigger number means more related), instead, we make 6 keywords all categorical. By using the Onehot encoding, we create 36 features to indicate different values for each keyword. The missing value will be handled differently. Because we treat the value as categorical, 3.5 (mean) will not be acceptable. We randomly choose from 3 or 4 to fill in the missing values from the original input feature.

Question 10 is the last feature that we need to handle, the question is asking to write a quote to this city. Because the test data is unseen and we can't access the true label, we can't train BOW while predicting. Instead, we only use the bag of words method to create a vocabulary list and new features to the original dataset based on the vocabulary list. The value of each word represents whether it is present in the quote of the sample (1 = present, 0 otherwise). We decided to use the fixed pre-generated vocabulary list to fit the model and process any new given data, because if any training processes are not allowed, the features and weight for each feature should be fixed. The main problem is that if the vocabulary list can't be generated based on the input dataset, some of the words in the original vocabulary list will never be present in the new dataset, and some words in the new dataset are also not in the vocabulary list. This possibly leads to many columns being all zeros, which means the feature is useless for prediction. We will try to use this strategy to train the model and compare the test accuracy. If it makes the model overfit, we will ignore question 10 while fitting the model.

In order to train the prediction model, we need to split the original dataset into training, validation and test sets. To make the choice of hyper-parameter generalize enough, we randomly shuffle the dataset before splitting it. Also, the data-splitting process will take care of transforming the type of dataset from Pandas data series to the NumPy array, which is more friendly for matrix calculation.

3 Model Evaluation and Exploration

requirement:(1 points) Model: A description of the model(s) that you are evaluating/exploring. We are expecting a thorough exploration of at least 3+ family of models, even if you don't ultimately use those models. We are looking for: – How you are applying these models. You don't need to reiterate what the models are and how they work. Instead, we're looking for a description of the choices you are making to apply the model to this task.

Our project aims to predict which city a person is thinking of, based on their subjective responses to a series of questions. We approached this challenge by exploring three distinct model families: Random Forest, Neural Network, and K-Nearest Neighbors (KNN), each offering unique strengths for handling the nuanced dataset compiled from the survey responses.

1. Random Forest

We chose the Random Forest model for its robustness in handling varied data types and its capability to manage overfitting.

Given our data's categorical nature and the presence of ordinal scales (e.g., city popularity ratings, architectural uniqueness), Random Forest's ensemble approach allowed us to capture complex patterns without being overly sensitive to noise.

We experimented with different numbers of trees and depth to find the optimal balance, focusing on minimizing bias while preventing overfitting, especially crucial due to the unseen nature of our test set.

Additionally,

2. Neural Network

Another option we considered for our prediction model was a Neural Network, as it allows for the updating of weights and biases through backpropagation during the training process. In our code, we implemented two approaches to construct Neural Networks for our prediction model. The initial method replicated the methods applied in Lab6, while the second approach utilizes the PyTorch library. Although these methods share similarities, they resulted in distinct models after the training process.

3. K-Nearest Neighbors (KNN)

We selected the K-Nearest Neighbors(KNN) algorithm for its simplicity and effectiveness in handling the classification problems, especially given the characteristics of our dataset, which includes a mix of continuous variables(e.g., the number of unique languages of a city) and categorical variables(e.g., rating the popularities of a city). One of the primary advantages of KNN is its intuitiveness: it classifies a given data point based on the majority class among its k-nearest neighbors. This approach is particularly beneficial for our project because it allows us to leverage the natural clustering of data points in the feature space, making it highly applicable to scenarios where relationships between data points are indicative of their categorization.

To optimize our KNN model, we focused on selecting the appropriate number of neighbors (k) and the distance meter used to identify these neighbours. The choice of k is significant to KNN performance. We experimented with various values of k to strike a balance between model complexity and generalization capability. A smaller k value makes the model sensitive to noise, leading to overfitting, while a larger k value might oversimplify the model, increasing bias. Similarly, we evaluated different distance metrics (Euclidean, Manhattan, and cosine similarity) and chose the most appropriate one for this project. (formula)Euclidean distance measures the distance between two data points. It suffers the curse of dimensity and is sensitive to data outliers, which means the big difference between data points will lead to a total different result. The other metric, Manhattan distance, calculates the sum of the absolute differences between data points. It is more offers robustness in higher dimensions with less sensitivity to outliers. Cosine similarity measures the cosine of the angle between two non-zero vectors in an inner product space, providing a measure of orientation similarity rather than magnitude. It is ideal for text analysis and frequency-based data where the orientation of the vectors is more important

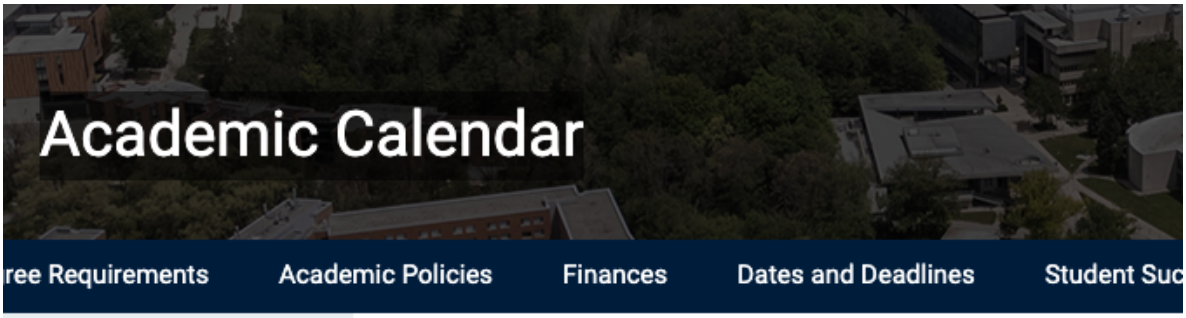

Figure 1: Enter Caption

than their magnitude and is effective in high-dimensional spaces. Considering our dataset, which contains almost 1500 features, the challenges of high dimensionality and outliers could significantly affect our model's performance. Thus, the metric less affected by these factors is more desirable. For each metric, we trained a KNN model and used it to make predictions on the test data. We also created confusion matrices to analyze the accuracy of our models."(confusion matrix analysis)

(Fix later)Our empirical evaluations, conducted through cross-validation, focused on measuring the model's precision and recall, given the importance of accurately classifying instances while minimizing false positives in our application context. The results indicated that our tuned KNN model achieves a commendable balance between sensitivity and specificity, making it a reliable choice for predicting [specific outcome based on your project].

4 Model Choice and Hyperparameters

using a consistent test set

(KNN)

4.1 Evaluation Metrics

5 Prediction

As the result, out prediction model would be the Neural Network with the number of hidden layer is

6 Workload Distribution