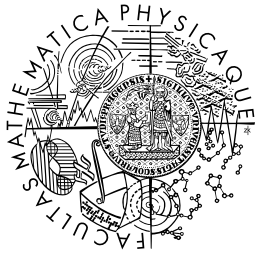

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Mitchell Borchers

Active learning in E-Commerce Merchant Classification using Website Information

Department of Theoretical Computer Science and Mathematical Logic

Supervisor of the master thesis: Mgr. Marta Vomlelová, Ph.D.
Study programme: Artificial Intelligence
Study branch: IUIPA

Prague 2023

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

Dedication.

Title: Active learning in E-Commerce Merchant Classification using Website Information

Author: Mitchell Borchers

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: Mgr. Marta Vomlelová, Ph.D., Department of Theoretical Computer Science and Mathematical Logic

Abstract: Data and the collection and analysis of data has become an important part of everyday life. For example, navigation, e-commerce, and social media all make use of immense amounts of data to provide users with suggestions on the best routes to take, which new items they might be most interested in, and which content might fit best with their interests. A variety of algorithms and methods exist to process the data and use it to make predictions. One such algorithm is xPAL, which is a decision-theoretic approach to measure the usefulness of a labeling candidate in terms of its expected performance gain. With xPAL and other active machine learning methods an optimal strategy can be explored to classify new data points.

Keywords: probabilistic active learning xPAL machine learning multi-class classification active learning

Contents

Introduction

One of the main challenges of creating a successful machine learning model is obtaining labeled data. With easy access to a variety of modern tools, devices, and sensors, we are able to rapidly collect unlabeled data. But, in supervised learning, prediction models are trained using labeled data. The problem is that acquiring labels for the collected data can be expensive, time-consuming, or even impossible in some cases.

However, methods have been developed to help reduce the number of labeled data required to train the classifier. Active learning is a semi-supervised machine learning framework where the model is trained with a smaller set of labeled data but which also aims to exploit trends within the unlabeled data. Active learning is a framework in which the learner has the freedom to select which data points are added to its training set (Roy and McCallum [2001]).

Active learning is different from other frameworks because it uses the unlabeled data and some evaluation criteria to determine which candidate could be the most beneficial to the model if it was given a label. In summary, the model requests the label from some oracle that provides the label then it takes this new labeled data point and rebuilds the classifier. We describe it as semi supervised active learning because of the oracle (typically a human) involved in the process that provides the label for the requested candidate data.

In our case we will provide a set of labeled data to the active learning framework (or sampling strategy). The sampling strategy will assume all the data is unlabeled and then choose a candidate from unlabeled data pool. Then the label is revealed and the classifier is updated using the new data point.

In our case we have some data (website urls) for some company or business that are given to us from our partner. From this data our partner currently utilizes human labor to browse the website and then label the url with a category (23 labels) and a sub-category (234+ tags) that branch from the main category but still have some relation. This is a repetitive and expensive task that could be automated using active learning.

To reduce the burden of human labeling we propose using a combination of tools, namely, Scrapy, Postgres, translation services, and semi supervised active learning that require occasional interaction where a human can label a candidate (if unlabeled) that is expected to be most beneficial to the classifier.

A website is required as input, then we use Scrapy to navigate to the webpage, collect and store the scraped data into the database. Next we access the data, translate the text, and add the translated data back into the database. We then create the model using the data from the database.

In the first section we introduce active learning and the different components of active learning. In the second and third sections we discuss the details of xPAL and the process of collecting and preparing the data, respectively. In the fourth section we look at combinations of different sampling strategies / classifiers and their performance results.

0.1 Notable Definitions

In this section we define some terms and ideas that will be helpful in understanding the upcoming sections.

Definition 1 (Beta Prior)**.** *A beta prior is a conjugate prior for the binomial distribution. It is a continuous probability distribution defined on the interval [0, 1] and is parameterized by two positive shape parameters, α and β. The beta distribution is defined as:*

$$Beta(\alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}$$

where Γ is the gamma function and x is a random variable.

Definition 2 (Conjugate Prior)**.** *A conjugate prior is a prior distribution that has the same functional form as the likelihood function. In other words, the posterior distribution will have the same functional form as the prior distribution.*

Definition 3 (Decision-Theoretic)**.** *Decision-theoretic active learning is a framework that uses the expected performance gain of a candidate to determine which candidate to label. The expected performance gain is the expected performance of the classifier after labeling the candidate minus the expected performance of the classifier before labeling the candidate. The expected performance of the classifier is the expected value of the performance measure given the posterior distribution of the classifier.*

Definition 4 (Dirichlet Distribution)**.** *The Dirichlet distribution is a multivariate generalization of the beta distribution. It is a continuous probability distribution defined on the K-simplex, $\Delta_K = \{x \in \mathbb{R}^K : x_i \geq 0, \sum_{i=1}^K x_i = 1\}$. The Dirichlet distribution is parameterized by a vector of positive shape parameters, $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_K)$. The Dirichlet distribution is defined as:*

$$Dir(\alpha) = \frac{\Gamma(\sum_{i=1}^K \alpha_i)}{\prod_{i=1}^K \Gamma(\alpha_i)} \prod_{i=1}^K x_i^{\alpha_i - 1}$$

where Γ is the gamma function and x is a random vector. The gamma function is defined as:

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$$

The gamma function is used as a normalizing constant to ensure that the probability density function integrates to 1 over the simplex, which is the space of all probability vectors that sum to 1.

Definition 5 (Ground Truth)**.** *Ground truth is the true label of a data point.*

Definition 6 (Posterior Probabilities)**.** *Posterior probability is a type of conditional probability that results from updating the prior probability with information summarized by the likelihood via an application of Bayes' rule. The posterior probability is the probability of an event occurring given that another event has occurred.*

Definition 7 (Omniscient Oracles)**.** *Omniscient oracle is a hypothetical entity that has complete knowledge of the true labels of all data points in a given dataset. An omniscient oracle knows the ground truth labels of all data points.*

1. Active Learning

1.1 Introduction

Russel and Norvig succinctly define an agent and different types of learning in their book "Artificial Intelligence: A Modern Approach" (Russell and Norvig [2009]), their definition is paraphrased here. They define an agent as something that acts and a rational agent as one that acts so as to achieve the best outcome. If there there is uncertainty, then the agent tries to achieve the best expected outcome. Any component of an agent can be improved by learning from data. The improvements and techniques used to make them depend on four major factors:

- Which component is to be improved.
- What prior knowledge the agent already has.
- What representation is used for the data and the component.
- What feedback is available to learn from.

Here we will mostly be focused on the final point, "What feedback is available to learn from". However, we will also discuss the importance of the second and third points because of our use of Bayesian learning and how the form and quality of the data affects the experiments. There are three main types of feedback that determine the three main types of learning which are unsupervised, reinforcement, and supervised learning.

In unsupervised learning an agent learns patterns even though no feedback is provided. In reinforcement learning, the agent learns from a series of rewards or punishments. In supervised learning, an agent learns from input-output pairs, which can be discrete or continuous, to find a function that maps the pairs.

The goal of supervised learning is given a training set of N example input-output pairs:

$$(x_1, y_1), (x_2, y_2), ...(x_N, y_N),$$

where each y_j was generated by some unknown function $y = f(x)$, find a function h that approximates the true function.

In reality, the lines separating the types of learning aren't so clear. Semi-supervised learning is also an important and widely used method. In semi-supervised learning we are given a few labeled examples that were labeled by some oracle (labeler, data annotator, etc.) and we must then make the most of a large collection of unlabeled examples. But what can we do with the unlabeled data?

Supervised learning models almost always get more accurate with more labeled data. Active learning is the process of deciding which data to select for annotation (Munro [2021]). In other words, the central component of an active learning algorithm is the selection strategy, or deciding which of the unlabeled data could be the most useful to the model if it was labeled. Active learning uses a selection

strategy that augments the existing classifier, it is not itself a classifier but rather a tool paired with a classifier.

Many strategies for choosing the next points to label exist. First we will discuss query functions then we will briefly define three basic sampling approaches: uncertainty, diversity, and random sampling to get an idea of sampling. We will then discuss some more advanced sampling approaches that are used in our experiments. When sampling the unlabeled data an ordered list is returned and the top candidate is the candidate that is expected to be most valuable for the model, but we are not strictly limited to taking just one candidate.

1.2 Query Function Construction

There are various techniques used to construct the querying functions we have discussed. We will focus on pool-based active learning, but a number of interesting and relevant ideas appear within other active-learning frameworks that are worth mentioning.

1.2.1 Pool-Based

The learner calculates the potential gain of all the unlabeled points in the pool, then requests the label for the point that maximizes the expected information gain for the classifier (Huang and Lin [2016]). For pool-based multiclass active learning, a labeled pool and an unlabeled pool are presented to the algorithm. In each iteration, the algorithm selects one instance from the unlabeled pool to query its label.

1.2.2 Stream-Based

The learner is provided with a stream of unlabeled points. On each trial, a new unlabeled point is drawn and introduced to the learner who must decide whether or not to request its label (Baram et al. [2004]). Note that the stream-based model can be viewed as an online version of the pool-based model.

1.2.3 Membership Queries

On each trial the learner constructs a point in input space and requests its label (Baram et al. [2004]). This model can be viewed as a pool-based game where the pool consists of all possible points in the domain.

1.3 Sampling Strategies

Sampling strategies, also referred to as selection strategies, are the core of the active learning process. The goal of sampling is to select the most useful data points from the unlabeled pool to label. The most useful data points are those that are expected to improve the classifier the most.

1.3.1 Random Sampling

Random sampling is self explanatory as it randomly selects an unlabeled data point from the pool and requests to have it labeled then used in the model.

1.3.2 Diversity Sampling

Diversity sampling is the set of strategies for identifying unlabeled items that are underrepresented or unknown to the machine learning model in its current state (Munro [2021]). The items may have features that are unique or obscure in the training data, or they might represent data that are currently under-represented in the model.

Either way this can result in poor or uneven performance when the model is applied or the data is changing over time. The goal of diversity sampling is to target new, unusual, or underrepresented items for annotation to give the algorithm a more complete picture of the problem space.

1.3.3 Uncertainty Sampling

Uncertainty sampling is the set of strategies for identifying unlabeled items that are near a decision boundary in the current machine learning model (Munro [2021]). Uncertainty sampling is simple given a classifier that estimates $P(C|w)$ (Lewis and Gale [1994]). On each iteration, the current version of classifier can be applied to each data point, and the data with estimated $P(C|w)$ values closest to 0.5 are selected, since 0.5 corresponds to the classifier being most uncertain of the class label.

These items are most likely to be wrongly classified, so they are the most likely to result in a label that differs from the predicted label, moving the decision boundary after they have been added to the training data and the model has been retrained.

1.3.4 PAL

Probabilistic Active Learning (PAL) follows a smoothness assumption and models for a candidate instance both the true posterior in its neighborhood and its label as random variables (Krempl et al. [2014]). By computing for each candidate its expected gain in classification performance over both variables, PAL selects the candidate for labeling that is optimal in expectation. PAL shows comparable or better classification performance than error reduction and uncertainty sampling, has the same asymptotic linear time complexity as uncertainty sampling, and is faster than error reduction.

1.3.5 xPAL

Extended probabilistic gain for active learning (xPAL) is a decision-theoretic selection strategy that directly optimizes the gain and misclassification error, and uses a Bayesian approach by introducing a conjugate prior distribution to determine the class posterior to deal with uncertainties (Kottke et al. [2021]).

Although the data distribution can be estimated, there is still uncertainty about the true class posterior probabilities.

These class posterior probabilities can be modeled as a random variable based on the current observations in the dataset. For this model, a Bayesian approach is used by incorporating a conjugate prior to the observations. This produces more robust usefulness estimates for the candidates.

1.3.6 ALCE

Active Learning with Cost Embedding (ALCE) is a non-probabilistic uncertainty sampling algorithm for cost-sensitive multiclass active learning (Huang and Lin [2016]). They first designed a cost-sensitive multiclass classification algorithm called cost embedding (CE), which embeds the cost information in the distance measure in a special hidden space by non-metric multidimensional scaling. They then use a mirroring trick to let CE embed the possibly asymmetric cost information in the symmetric distance measure.

1.3.7 QBC

Query by committee uses an ensemble of classifiers that are trained on bootstrapped replicates of the labeled set (Seung et al. [1992]). The idea is to train a committee of classifiers on the available labeled data and then use the committee to select the most informative unlabeled data for labeling (Freund et al. [1997]). The committee consists of several classifiers, each trained on a slightly different subset of the available labeled data.

The QBC algorithm measures the disagreement of the committee's predictions on each unlabeled data point. The intuition is that if the committee members disagree then it is likely to be a difficult data point for the current classifier and thus informative for labeling.

The algorithm selects a fixed number of the most informative examples and asks the user or oracle to label them. The labeled examples are then added to the labeled dataset, and the committee is retrained on the expanded labeled dataset. This process is repeated until the algorithm achieves a desired level of accuracy or the available labeling budget is exhausted.

1.3.8 EER

Monte Carlo estimation of error reduction (EER) estimates future error rate by log-loss, using the entropy of the posterior class distribution on a sample of the unlabeled examples, or by 0-1 loss, using the posterior probabilities of the most probable class for the sampled unlabeled examples (Roy and McCallum [2001]).

1.4 Summary

Now it should be more clear how the sampling strategy is the major component of active learning. The query function construction is also important but it is just a means of routing the data to be sampled. In the next chapter we will look into the specifics of xPAL.

2. Understanding xPAL

We have introduced many different active learning models in the previous section, and we will test some of these models on our data. However, we will mainly focus on using the xPAL sampling strategy and a pool based query function. The xPAL sampling strategy is a decision-theoretic approach to measure the usefulness of a labeling candidate in terms of its expected performance gain (Kottke et al. [2021]). We can estimate the data distribution but we are uncertain about the true class posterior probabilities. The class posterior probabilities are modeled as a random variable based on the current observations. Therefore a Bayesian approach is used by incorporating a conjugate prior to the observations. In general, the idea is to estimate the expected performance gain for the classifier, using the unlabeled data, and then select the best data point and request its label. Variable descriptions used in the following equations and explanations are listed in Table 2.1

	Descriptions
C	Number of classes
x	Input $x \in \mathbb{R}^n$
y	Output $y \in l_1, ..., l_C$
L	Loss ??
R	Risk $R(f^{\mathcal{L}}) \in \mathbb{R}_0^x$
$R_{\mathcal{E}}$	Empirical risk ??
$\mathcal{L}$	Set of labeled data $\{(x_1, y_1), ..., (x_n, y_n)\}$
$\mathcal{U}$	Set of unlabeled data $\{x_1, ..., x_n\}$
$\mathcal{E}$	Set of labeled and unlabeled data
$p(x, y)$	Joint distribution of random variables x and y
$f^{\mathcal{L}}$	Classifier that maps input x to output y

Table 2.1: Variable names and definitions.

2.1 Kernel

A kernel based classifier is used in xPAL which determines the similarity of two data points. The kernel function $\boldsymbol{K}(x, x')$ is a function that maps two data points to a real number. The kernel frequency estimate $\boldsymbol{k}_x^{\mathcal{L}}$ of an instance $\boldsymbol{x}$ is calculated using the labeled instances $\mathcal{L}$. The y-th element of that C-dimensional vector describes the similarity-weighted number of labels of class y.

$$\boldsymbol{k}_x^{\mathcal{L}}, y = \sum_{(x',y') \in \mathcal{L}} \mathbb{1}_{y=y'} \boldsymbol{K}(x, x') \tag{2.1}$$

The Parzen Window Classifier uses the labeled data for training and predicts the most frequent class and was selected by Kottke et al. to use because of its speed and ability to implement different kernels depending on the data (Kottke et al. [2021]). It was used for all the selection strategies in their experiments.

$$f^{\mathcal{L}}(x) = \arg\max_{y \in \mathcal{Y}} \left(\boldsymbol{k}^{\mathcal{L}}_{x,y} \right). \tag{2.2}$$

We will mainly use the PWC classifier in our experiments but we will also evaluate other classifiers and compare their performance.

2.2 Risk

For xPAL, Kottke et al. use the classifications error as the performance measure and minimize the zero-one loss. The risk describes the expected value of the loss relative to the joint distribution given some classifier. The zero-one loss returns 0 if the prediction from the classifier is equal to the true class else it returns 1. The risk is a theoretical concept that cannot be computed directly since it requires knowledge of the entire population distribution. Instead, we typically try to approximate the risk using the empirical risk.

$$R(f^{\mathcal{L}}) = \mathop{\mathbb{E}}_{p(x,y)} [\boldsymbol{L}(y, f^{\mathcal{L}}(x))] \tag{2.3}$$

$$= \mathop{\mathbb{E}}_{p(x)} \left[\mathop{\mathbb{E}}_{p(y|x)} [\boldsymbol{L}(y, f^{\mathcal{L}}(x))] \right] \tag{2.4}$$

$$\boldsymbol{L}(y, f^{\mathcal{L}}(x)) = \mathbb{1}_{f^{\mathcal{L}}(x) \neq y} \tag{2.5}$$

Because it is not known how the data is generated Kottke et al. use a Monte-Carlo integration with all the data $\mathcal{E}$ to represent the generator. The empirical risk $R_{\mathcal{E}}$ is the average of the loss over all the data points in the dataset. It refers to the average value of a given loss function over a finite set of observed data points. The empirical risk is a computable quantity that can be used as an estimate of the risk. However, it is only an approximation and is subject to sampling error.

$$R_{\mathcal{E}}(f^{\mathcal{L}}) = \frac{1}{|\mathcal{E}|} \sum_{x \in \mathcal{E}} \mathop{\mathbb{E}}_{p(y|x)} \left[\boldsymbol{L}(y, f^{\mathcal{L}}(x)) \right] \tag{2.6}$$

$$= \frac{1}{|\mathcal{E}|} \sum_{x \in \mathcal{E}} \sum_{y \in \mathcal{Y}} p(y|x) \boldsymbol{L}(y, f^{\mathcal{L}}(x)) \tag{2.7}$$

2.3 Conjugate Prior

The conditional class probability $p(y|x)$ depends on the ground truth which is unkown. As a result the conditional class probability is exactly the y-th element of the unknown ground truth vector $\boldsymbol{p}$. The nearby labels from $\mathcal{L}$ can be used to estimate the ground truth $\boldsymbol{p}$ becuase the oracle provides the labels according to p. If we assume a smooth distribution then the estimate is relatively close to the ground truth if we have enough labeled instances.

$$p(y|x) = p(y|t(x)) = p(y|\boldsymbol{p}) = \mathrm{Cat}(y|\mathbf{p}) = p_y \tag{2.8}$$

A Bayesian approach is used for estimation by calculating the posterior predictive distribution (calculating the expected value over all possible ground truth

values). The probability of y given some x is approximately equal to the kernel frequency estimate of x.

$$p(y|x) \approx p(y|\boldsymbol{k}_x^{\mathcal{L}}) = \mathop{\mathbb{E}}_{p(p|\boldsymbol{k}_x^{\mathcal{L}})}[p_y] = \int p(p|\boldsymbol{k}_x^{\mathcal{L}}) p_y dp \tag{2.9}$$

Bayes theorem is then used to determine the posterior probability of the ground truth at instance x in Equation 2.10. The likelihood $p(\boldsymbol{k}_x^{\mathcal{L}}|p)$ is a multinomial distribution because each label has been drawn from $Cat(y|p)$. A prior is introduced and selected as a Dirichlet distribution with $\alpha \in \mathbb{R}^C$ as this is the conjugate prior of the multinomial distribution. An indifferent prior is choosen and each element of alpha is set to the same value. The Dirichlet distribution is an analytical solution for the posterior when the conjugate prior of the multinomial likelihood are used.

$$p(p|\boldsymbol{k}_x^{\mathcal{L}}) = \frac{p(\boldsymbol{k}_x^{\mathcal{L}}|p)p(p)}{p(\boldsymbol{k}_x^{\mathcal{L}})} \tag{2.10}$$

$$= \frac{\text{Mult}(\boldsymbol{k}_x^{\mathcal{L}}|p) \cdot \text{Dir}(p|\alpha)}{\int \text{Mult}(\boldsymbol{k}_x^{\mathcal{L}}|p) \cdot \text{Dir}(p|\alpha) dp} \tag{2.11}$$

$$= \text{Dir}(p|\boldsymbol{k}_x^{\mathcal{L}} + \alpha) \tag{2.12}$$

The conditional class probability is determined next from Equation 2.9. It is calculated with the expected value of the Dirichlet distribution.

$$p(y|\boldsymbol{k}_x^{\mathcal{L}}) = \mathop{\mathbb{E}}_{\text{Dir}(p|\boldsymbol{k}_x^{\mathcal{L}}+\alpha)}[p_y] \tag{2.13}$$

$$= \int \text{Dir}(p|\boldsymbol{k}_x^{\mathcal{L}} + \alpha) p_y dp \tag{2.14}$$

$$= \frac{(\boldsymbol{k}_x^{\mathcal{L}} + \alpha)_y}{||\boldsymbol{k}_x^{\mathcal{L}} + \alpha||_1} \tag{2.15}$$

The last term is the y-th element of the normalized vector. The 1-norm is used to normalize the vector.

2.4 Risk Difference Using the Conjugate Prior

Next, we insert equation 2.15 into the empirical risk equation 2.7. We are approximating $p(y|x)$ with $p(y|\boldsymbol{k}_x^{\mathcal{L}})$ which is the empirical risk based on the labeled data $\mathcal{L}$.

$$\hat{R}_{\mathcal{E}}(f^{\mathcal{L}}, \mathcal{L}) = \frac{1}{|\mathcal{E}|} \sum_{x \in \mathcal{E}} \sum_{y \in \mathcal{Y}} \frac{(\boldsymbol{k}_x^{\mathcal{L}} + \alpha)_y}{||\boldsymbol{k}_x^{\mathcal{L}} + \alpha||_1} \cdot L(y, f^{\mathcal{L}}(x)) \tag{2.16}$$

Now lets assume we add a new labeled candidate (x_c, y_c) to the labeled data set $\mathcal{L}$. We will now denote the set with the newly labeled data point $\mathcal{L}^+ = \mathcal{L} \cup \{(x_c, y_c)\}$. Next we need to determine how much this new data point improved our classifier. We then make an estimate of the gain in terms of risk difference using the probability to estimate the ground truth.

$$\Delta\hat{R}_{\mathcal{E}}(f^{\mathcal{L}^+}, f^{\mathcal{L}}, \mathcal{L}^+) = \hat{R}_{\mathcal{E}}(f^{\mathcal{L}^+}, \mathcal{L}^+) - \hat{R}_{\mathcal{E}}(f^{\mathcal{L}}, \mathcal{L}^+) \tag{2.17}$$

$$= \frac{1}{|\mathcal{E}|} \sum_{x \in \mathcal{E}} \sum_{y \in \mathcal{Y}} \frac{(\boldsymbol{k}_x^{\mathcal{L}^+} + \alpha)_y}{||\boldsymbol{k}_x^{\mathcal{L}^+} + \alpha||_1} \cdot \Big(L(y, f^{\mathcal{L}^+}(x)) - L(y, f^{\mathcal{L}}(x)) \Big) \tag{2.18}$$

The observations used to estimate the risk are the same for both the old and new classifiers. We do this because we assume that adding labeled data will make the classifier better, so this allows us to more accurately compare the current classifier and the new one.

2.5 Expected Probabilistic Gain

If we are able to reduce the error with the new $\mathcal{L}^+$ model then equation 2.18 will be negative. As a result, we negate this term and maximize the expected probabilistic gain. To simplify we set $\alpha = \beta$.

$$\text{xgain}(x_c, \mathcal{L}, \mathcal{E}) = \mathop{\mathbb{E}}_{p(y_c|\boldsymbol{k}_{x_c}^{\mathcal{L}})} \Big[-\Delta\hat{R}_{\mathcal{E}}(f^{\mathcal{L}^+}, f^{\mathcal{L}}, \mathcal{L}^+ \Big] \tag{2.19}$$

$$= -\sum_{y \in \mathcal{Y}} \frac{(\boldsymbol{k}_x^{\mathcal{L}} + \beta)_y}{||\boldsymbol{k}_x^{\mathcal{L}} + \beta||_1} \cdot \frac{1}{|\mathcal{E}|} \sum_{x \in \mathcal{E}} \sum_{y \in \mathcal{Y}} \frac{(\boldsymbol{k}_x^{\mathcal{L}^+} + \alpha)_y}{||\boldsymbol{k}_x^{\mathcal{L}^+} + \alpha||_1} \cdot \Big(\boldsymbol{L}(y, f^{\mathcal{L}^+}(x)) - \boldsymbol{L}(y, f^{\mathcal{L}}(x)) \Big) \tag{2.20}$$

2.6 xPAL Selection Strategy

The xPAL selection strategy chooses this candidate $x_C^* \in \mathcal{U}$ where the gain is maximized:

$$x_c^* = \arg\max_{x_c \in \mathcal{U}} \left(\text{xgain}(x_c, \mathcal{L}, \mathcal{E}) \right) \tag{2.21}$$

3. Data Review

In this chapter we take a deeper look into the data and the process of collecting, translating, and encoding the data. Our partner has provided a small sample of 1000 labeled data points. This data was manually labeled by an human annotator.

3.1 Overview

The data consists of a merchant name, merchant website (url), merchant category, and merchant tag as shown in Table 3.1.

merchant name	merchant url	merchant category	merchant tags
State Hospital	http://hospital.com/	Health	'{"Clinic"}'

Table 3.1: This is an example of a single data point from the original data set.

The current process consists of giving the merchant url to an annotator and the annotator then views the website and either can instantly provide a label and tags for the website or in some cases may need to browse further into the website (by viewing sibling pages such as the 'About Us' sections or product pages) to get an idea of how the website should be classified.

The merchant tags are ordered by specificity, with the first tag in the list being the most general and the final being the most specific. An example of the tag hierarchy is show in Table. 3.2 where we can see that this sample consists of data from various categories all contained within the 'Eco' side tag grouping.

Category	Level 1 Tag	Level 2 Tag	Level 3 Tag	Side Tag
Travel	Local Transport	Micro-mobility	Bike Sharing	Eco
		Public Transport		Eco
Fashion	Clothing - Other	Second Hand		Eco
Car	Charging Station			Eco
	Car Sharing			Eco

Table 3.2: This is an example of how the tags use different levels.

The tags are important because they allow us to separate the data even further and group/ sort the data differently. However, in our work we didn't opt to use the tags at this time.

3.2 Collection

Similarly to the annotator our goal is to automate the navigation, collection/ storing process, and classification of the website. This pipeline speeds up the browsing process and can allow the annotator to spend much less time annotating and require the annotator to only annotate data expected to drastically improve the classifier.

The initial 1000 data points we received were labels with a pointer (a url) to where the text data is located. The labels needed the text from the websites

that the annotator viewed to begin the classifying process. To gather the text data from the websites we used the Scrapy framework to extract text data from a single top level page of a website. We chose only to scrape the top level (main or home) page text because of the results published in another study where it was observed that adding more pages to the data set does not necessarily mean obtaining better results (Sahid et al. [2019]).

Out of these initial data points 179 contained links that could not be accessed or links that provided no text data that could be scraped. Out of the remaining 821 data points 274 of them were in English.

3.3 Processing

It is important for us to have the data in English as it allows us to exploit stop words when using the Scikit-Learn TF-IDF vectorizer to construct our data set. Stop words are words like “and”, “the”, “him”, which are presumed to be uninformative in representing the content of a text, and which may be removed to avoid them being construed as signal for prediction (Fabian Pedregosa et al. [2023]). This holds true for our data set as well.

Out of the remaining 275 English data points the data was distributed into the categories as shown in Figure 3.1.

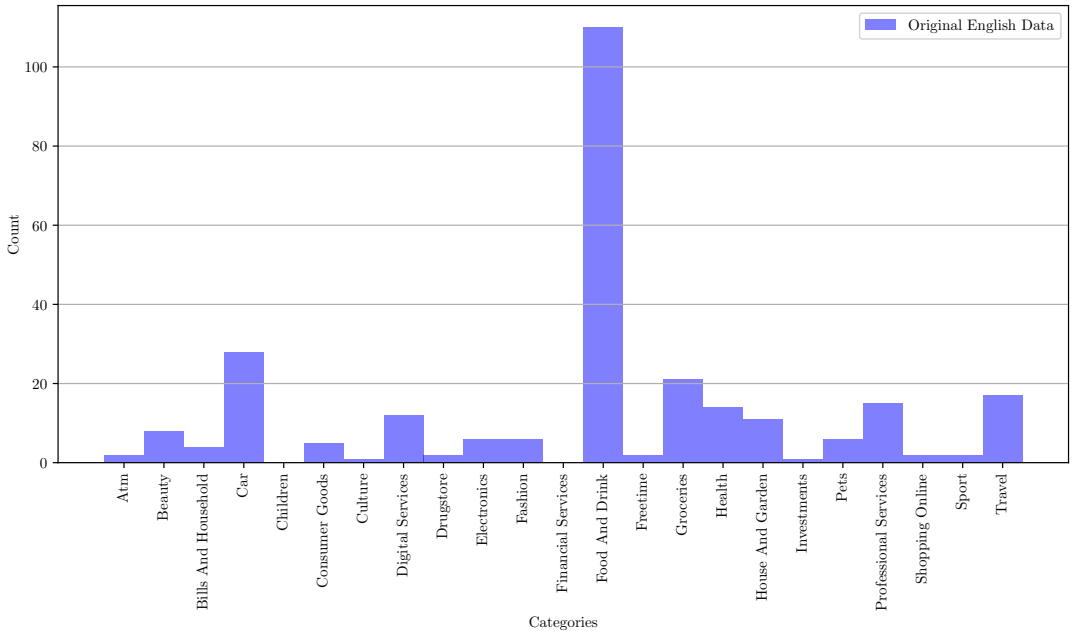

Figure 3.1: The histograms for the original usable english data.

At this stage, it was clear that our data set wasn’t representing all categories equally. The ’Food and Drink’ category has many more data points then the ’Culture’ and ’Investments’ categories which each had a single data point. The ’Children’ and ’Financial Services’ categories weren’t represented at all. Obviously this was problematic because we would like to have, minimum, three data points in each category to build train, test, and validation data sets.

At this point we had a significant amount of data that wasn't being used. We decided to find a way to translate the existing data. We tried various libraries available on GitHub but we weren't getting good results. We found that Azure had a service available and a free option of up to 2 million characters translated per month. This was a viable option and we were able to use this API to translate the remaining data. We limited the number of characters to 1000 per data point from the scraped text to avoid maxing out the API.

In addition to the original data we also manually collected and labeled 141 additional data points. All the original data and additional data are shown in Figure 3.2 and discretely in Appendix A in Table A.1.

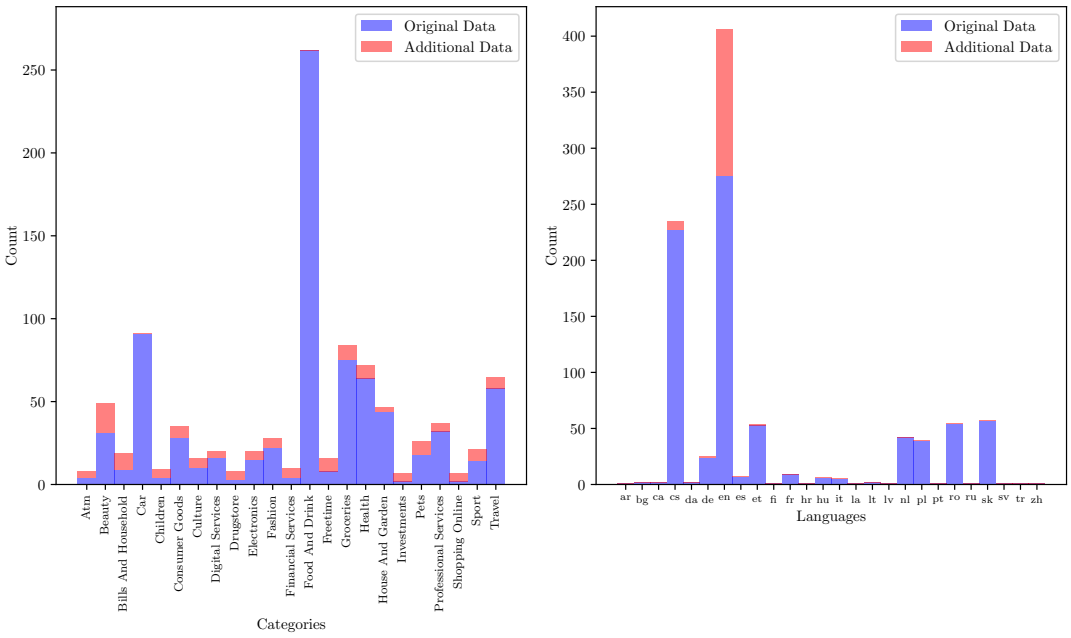

Figure 3.2: The histograms for the original and additional data for all languages.

After translating the data we used the TF-IDF vectorizer from Scikit-Learn. We were able to find highly correlated words for each category using Chi Squared analysis with only the original data, shown below in Table 3.3. Some categories such as 'Culture', 'Digital Services', 'Shopping Online' that have few data points have words such as 'kihnu', 'td', 'patria', respectively, which have no relative meaning to the category in English.

From what we discussed in the previous section, we can see that the 'Culture' category has only one data point and the 'Digital Services' category has only two data points. This is problematic because if the single data point we have is doesn't represent the category well then we will continue to have difficulty classifying until we have more robust data.

We also calculated the variable importance using the RandomForestRegressor from Scikit-Learn and provided a list of the top 20 most important words from the TF-IDF vectorizer. This helps us orient ourselves within the data as well as check if there may be any anomalies. The list of top 20 most important words are shown in Table A.2 in the Appendix.

Table 3.3: Keywords from TF-IDF with Chi Squared using the original data.

	Keyword 1	Keyword 2	Keyword 3
Atm	banking	zu	bank
Beauty	hairdressing	hairdresser	hair
Bills And Household	recycling	liberty	internet
Car	stations	auto	car
Children	toy	sold	toysrus
Consumer Goods	wallpaper	flowers	flower
Culture	museum	theater	kihnu
Digital Services	td	bitly	servers
Drugstore	alert	enable	detergent
Electronics	electronic	onoff	computers
Fashion	jewellery	men	women
Financial Services	pre	nissan	insurance
Food And Drink	bar	cafe	restaurant
Freetime	searched	casino	smartflex
Groceries	functioning	liquor	bakery
Health	optics	dental	pharmacy
House And Garden	wallpapers	paints	hardware
Investments	strive	investment	patria
Pets	breeding	pet	veterinary
Professional Services	faculty	parcel	school
Shopping Online	web	owner	joom
Sport	functional	singltrek	adidas
Travel	rooms	accommodation	hotel

4. Testing

4.1 Original Data

Here we will explore the results of different classifiers and active learning strategies on the original data. The original data consisted of data shown previously in Figure 3.1 and discretely in Table A.1 in Appendix A.

4.1.1 Active Learning with PWC and RBF Kernel

In Figure 4.1 we have the train and test errors for four different active learning sampling strategies and we can see that xPAL and PAL both perform relatively well before starting to over-fit the training data. We reviewed the resulting error data and xPAL outperformed PAL by about 1.5%. However, xPAL has much faster running time compared to PAL.

We weren't satisfied with the testing error which leveled out to about 70% for each sampling strategy. PAL and xPAL were able to rapidly reduce the testing error early on in the training process while random selection and QBC weren't able to determine the data with the highest information gain.

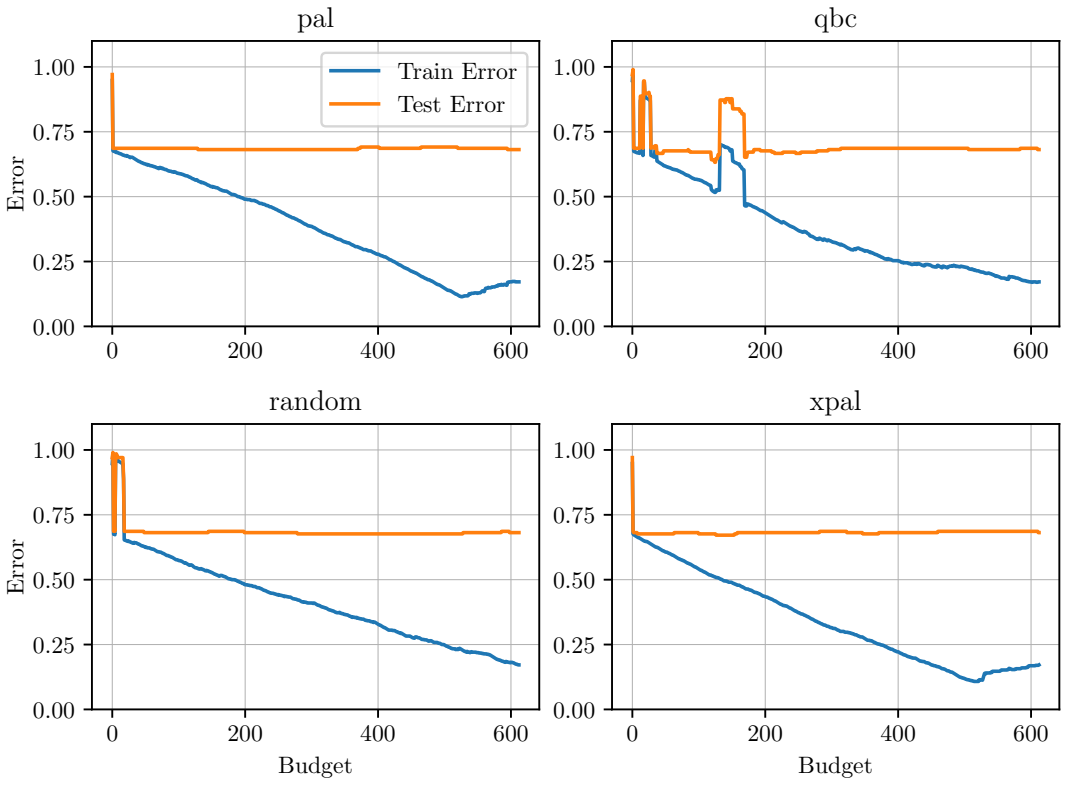

Figure 4.1: Train and test error using different query strategies and RBF kernel for the PWC classifier.

The radial bias function (RBF) kernel is a popular kernel function. It is defined as:

$$K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) \tag{4.1}$$

where σ is a parameter that controls the smoothness of the kernel and x_i and x_j are the two points in the feature space to compare. As seen in the Figure 4.1 when the PWC classifier uses the RBF kernel it doesn't perform well with this data.

4.1.2 Active Learning with PWC and Cosine Kernel

In Figure 4.2 we have the train and test errors for the same four different active learning sampling strategies tested on the same data. The only change was that we used Cosine kernel instead of the RBF kernel.The Cosine kernel is another important kernel function that is used in many machine learning algorithms. It is defined as:

$$K(x_i, x_j) = \frac{x_i \cdot x_j}{\|x_i\| \, \|x_j\|} \tag{4.2}$$

where x_i and x_j are the two points in the feature space to compare. We found that using the Cosine kernel reduced the test error across the board by ∼15%.

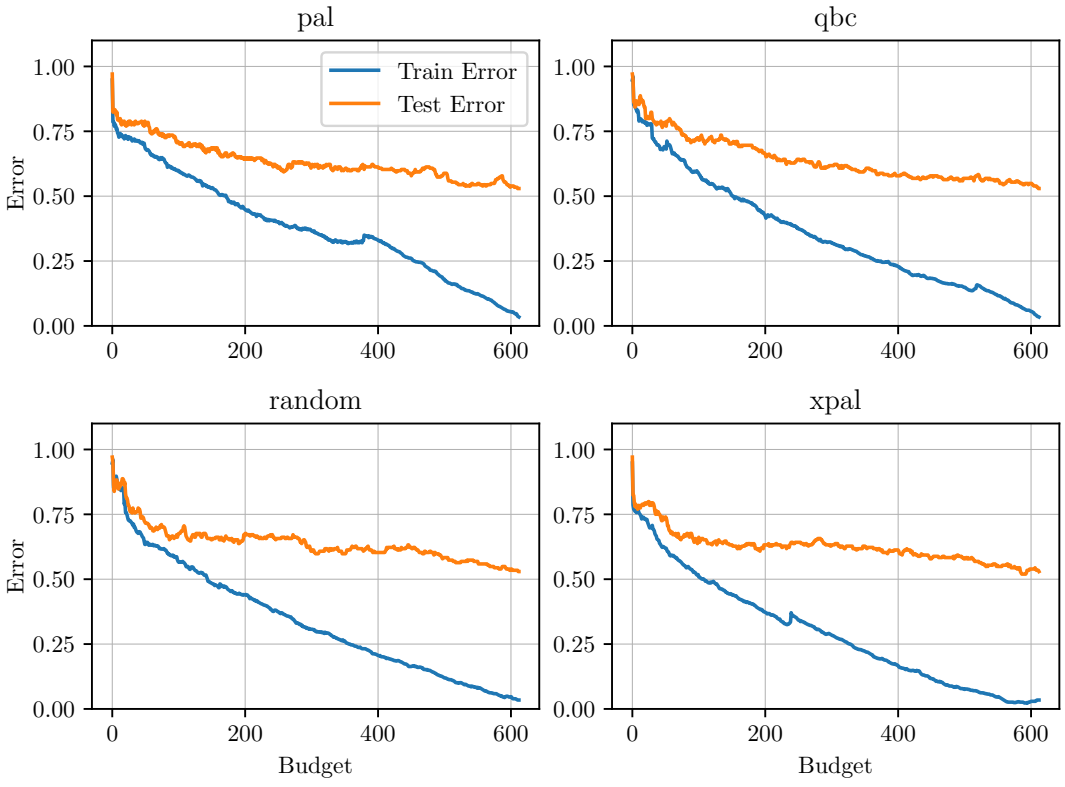

Figure 4.2: Train and test error using different query strategies and Cosine kernel for the PWC classifier.

In Figure 4.2 PAL and xPAL were able to reduce the training error, by about 20% and 25% respectively, early in the training process compared to random

selection and QBC. We also tested the other sampling strategies with the Cosine kernel and found that the results were similar. The other sampling strategies and their test data results are shown in Table 4.3 along with the test data from Figure 4.2.

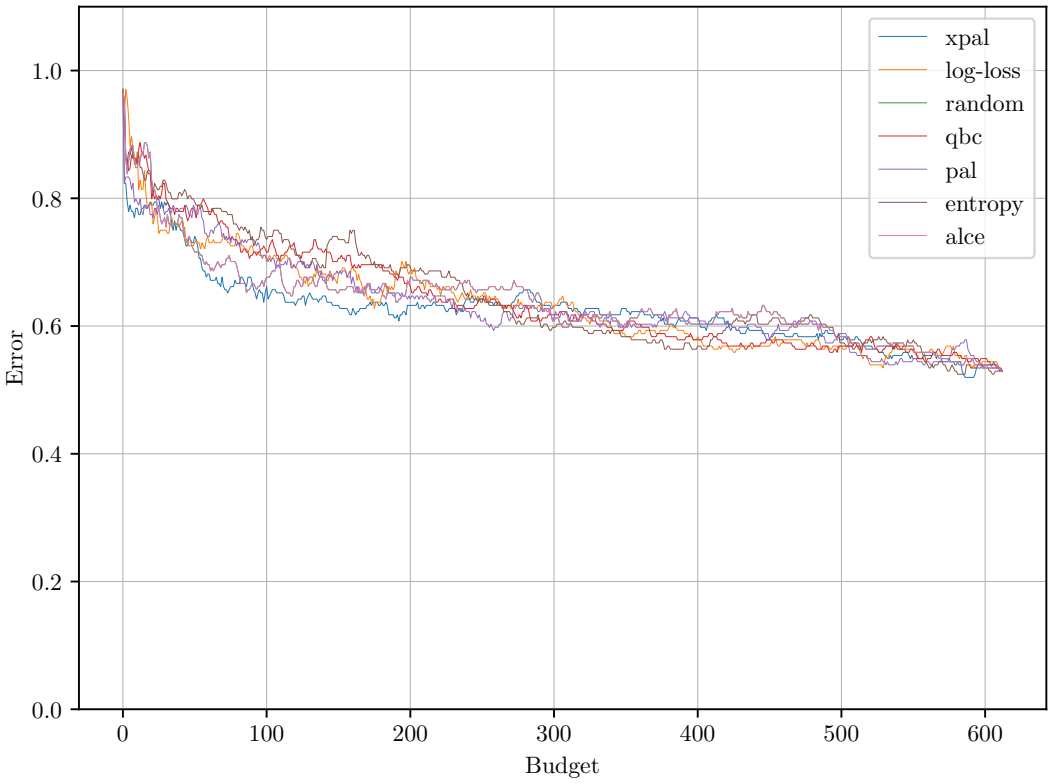

Figure 4.3: Comparing test error using different query strategies and Cosine kernel for the PWC classifier.

We can see that the sampling strategies test performance converges over time (as we are using the same data and classifier) but xPAL has the appears to have an absolute minimum near the 600 budget mark in comparison to all sampling strategies. XPAL also appears to be performing well early on in the training process, in the 100-200 budget range.

Figure 4.3 shows that xPAL seems to be performing the best with our data but we wanted to see if we ran more tests with different train-test splits how the results would average out and which sampling strategy would perform the best. We ran 10 different data splits with each of the 7 sampling strategies and then took the average to get a smoother curve compared to the single run results shown in Figure 4.3. The results for this experiment are shown in Figure 4.4.

4.1.3 Scikit-Learn Classifier Evaluation

We also decided to try out the boilerplate classifiers from Scikit-Learn to compare performances. Again we used the original data with the same TF-IDF vectorizer as used with the previous active learning models to stay consistent. Cross validation was also used here but was not used in the previous sections. We decided to

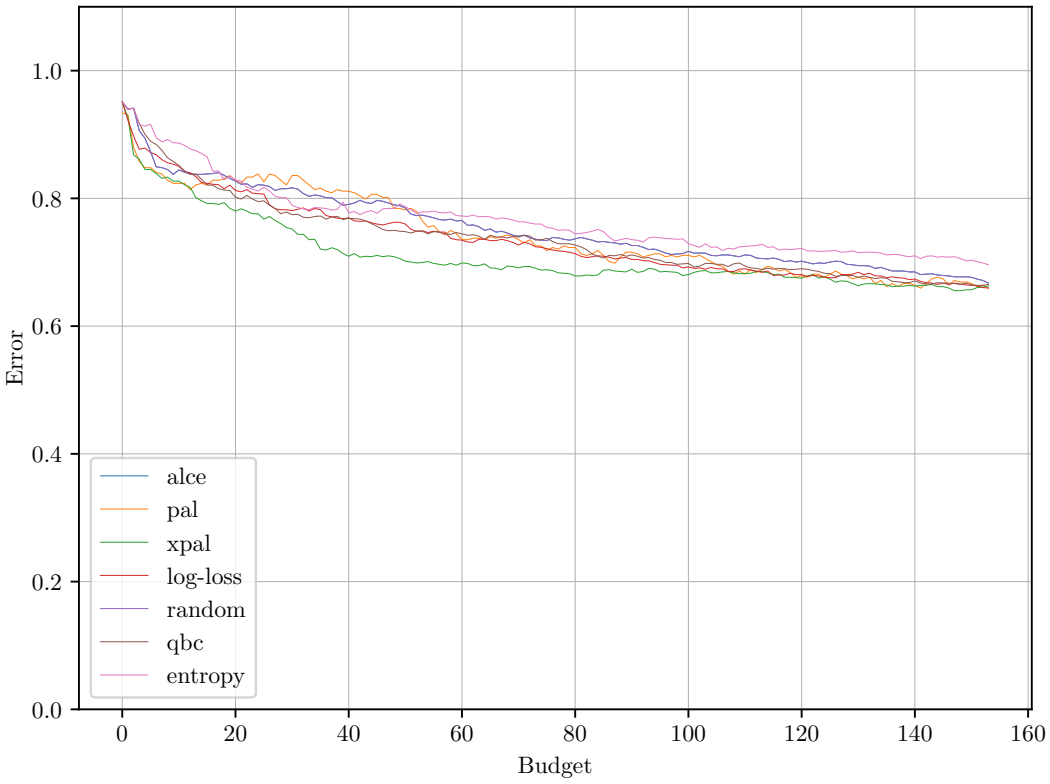

Figure 4.4: Comparing test error using different query strategies and Cosine kernel for the PWC classifier.

use the Cosine kernel when using LinearSVC as we learned that the performance increased in comparison to the RBF kernel when using PWC.

The results for a variety of different classifiers are shown in Figure 4.5. In the box-plot, the whiskers extend from the box to the furthest data points that are within 1.5 times the inter-quartile range (IQR) of the box. Any data points that are beyond the whiskers are considered outliers and are plotted as individual points or symbols (diamonds) as seen in Figure 4.5.

The LinearSVC classifier performed best compared to other classifiers and it is a fast running algorithm even with data that has a large feature set. We decided to look further into LinearSVC and create a few models to evaluate its performance. We created three models, the first was a boilerplate LinearSVC with no argument modifications, the second model used the class weights parameter set to 'balanced'. The 'balanced' mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as $n\_samples/(n\_classes * np.bincount(y))$.

For the third test we created a dictionary of weights for each class using the Cosine decay function. The weights for each category ranged from 0.1 to 1.0 where the most frequent classes had smaller weights. The results are shown in Table 4.1. The Cosine decay function is defined as:

$$w_i = \frac{1}{2}\left(1 + \cos\left(\frac{\pi t}{T}\right)\right) \tag{4.3}$$

where w_i is the weight for the i^{th} class, t is the current iteration, and T is the

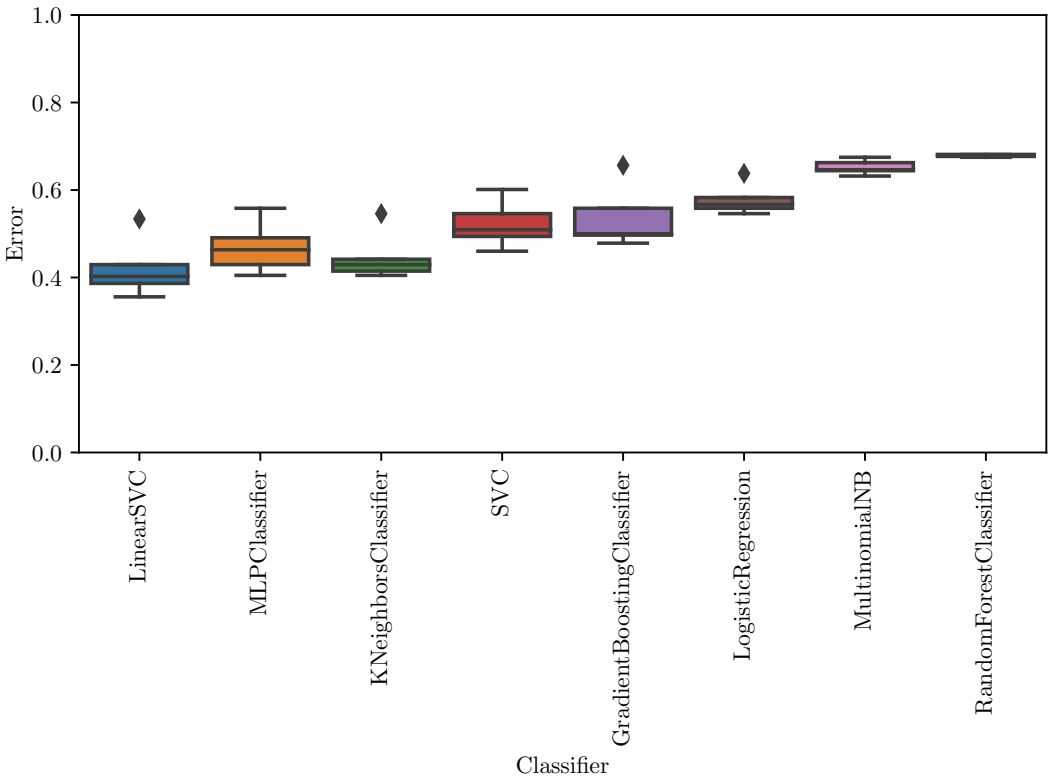

Figure 4.5: Performance of standard Scikit-Learn classifiers without optimization.

total number of iterations. The Cosine decay function is a common function used for weights in machine learning algorithms.

Table 4.1: Error for three differing LinearSVC models.

Model	Error
Cosine Decay Weights	0.392
Boilerplate	0.407
Balanced Weights	0.441

We also tested some other models that we expected to perform well, namely K-Nearest Neighbors and Neural Networks. The results are shown in Table 4.2. We can see that the K-Nearest Neighbors classifier and the Tensor Flow Neural Network classifier performed slightly worse compared to LinearSVC.

Table 4.2: Keywords from TF-IDF with Chi Squared using the original data.

Model	Error
LinearSVC	0.392
Neural Network	0.446
KNN	0.451

We attempted to boost performance from the LinearSVC classifier using GridSearchCV and Bagging but were not able to make significant improvements in

performance from what we observed in Table 4.2. The precision-recall curve is shown in Figure 4.6 and the confusion matrix is shown in Figure 4.7 for the best performing LinearSVC classifier.

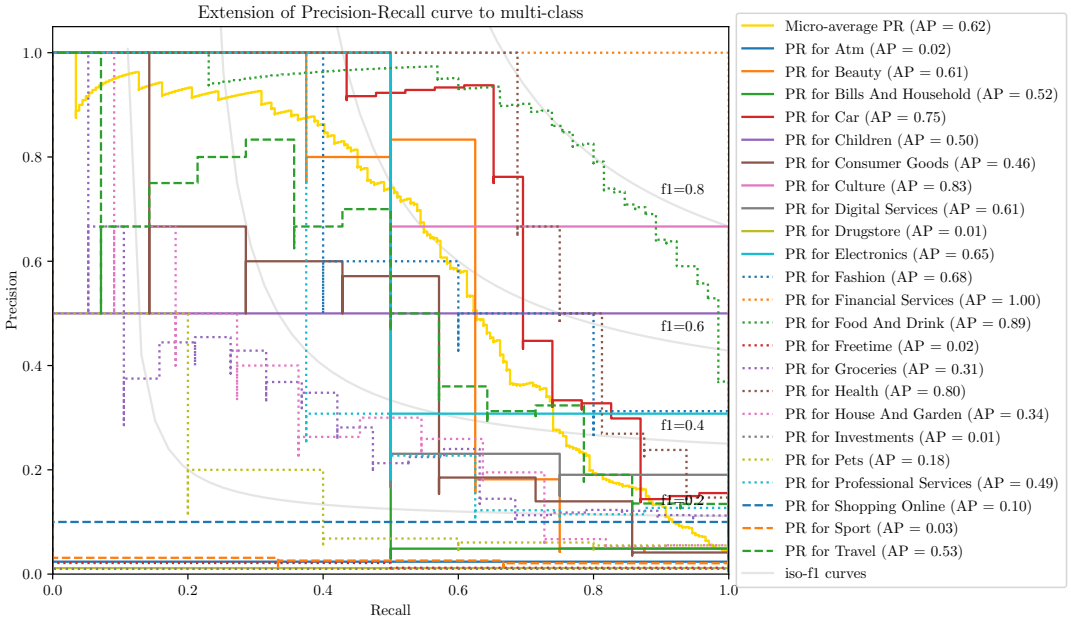

Figure 4.6: Precision-recall curve for the best performing LinearSVC classifier.

It is important to note that the precision-recall curve is not a perfect metric for this problem as the classes are not balanced. We can see this imbalance clearly in Figure 4.6 where we have straight lines and clear divisions. This is a result of having a small number of data in a class. However, we can also see that for some classes the precision is very high even though we have very few data points. Here we are namely concerned with the 'Culture' and 'Beauty' categories which have 10 and 31 data points respectively.

The confusion matrix may be a better metric for visualizing this problem as it shows the number of true positives, false positives, true negatives, and false negatives.

4.2 Original and Additional Data

In this section we will explore the results of different classifiers and active learning strategies on the original data plus the additional data.

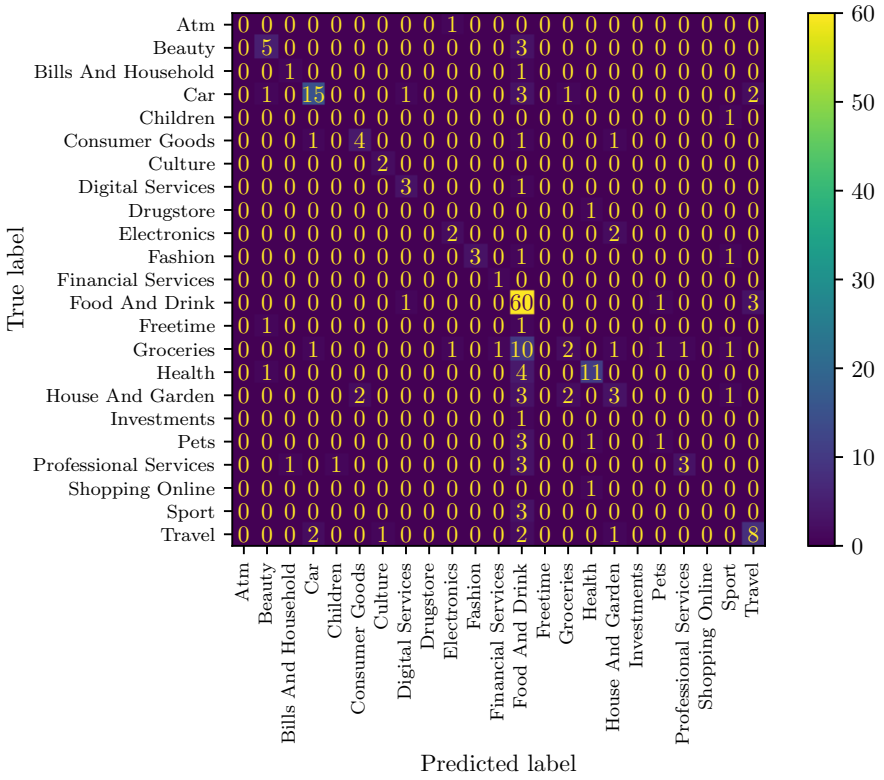

Figure 4.7: Confusion matrix for the best performing LinearSVC classifier.

Conclusion

The entire process of data collection, storage, retrieval, processing, and testing allowed us to create a customized pipeline and explore active learning using website text data to classify online merchants and other websites. The collected data wasn't perfect but we were able to use the data to create a model that could classify websites into 1 of 23 categories with ∼60% accuracy using a linear support vector classifier.

Pairing this classifier with xPAL allows us to create a pipeline to choose the most informative data points to add to the training set and reduces the amount of time it takes to train the model as well as the size of the classifier.

Many sites had multiple languages present on the websites and this could be seen in our scraped data. This posed a challenge for us because when using the API for translation we were making it more difficult to detect the language but also adding non english words into the TF-IDF. Solving this issue could potentially help improve the performance of the classifier.

Improvements

To improve the results of the active learning process we would suggest possibly making a number of changes. One of the first changes would be to find a better way to profile a website and make sure the quality of the text data from the websites were better. For example, a stronger web scraper would have allowed us to avoid potential IP address restriction issues and scrape data from social media websites that refused to allow our scraper to collect any data and resulted in some unusable data. We could have also run our own tests regarding how the number of sibling pages could have fortified the data.

To be extremely thorough, we could have run exhaustive tests for *all* the available classifiers within Scikit-Learn using GridSearchCV and other ensemble methods to see if any more performance gains were attainable.

Moving Forward

Next steps could include constructing an industrial web scraper. The scraper could be designed to scrape multiple pages of a website, scrape social media pages, and possibly scrape other websites that are linked to the original website. This could improve the quality of the data and allow for more accurate classification.

Bibliography

Yoram Baram, Ran El Yaniv, and Kobi Luz. Online choice of active learning algorithms. *Journal of Machine Learning Research*, 5(Mar):255–291, 2004.

Alexandre Gramfort Fabian Pedregosa, Gael Varoquaux, Vincent Michel, et al. 6.2. Feature extraction — scikit-learn.org. `https://scikit-learn.org/stable/modules/feature_extraction.html`, 2023. [Accessed 03-Mar-2023].

Yoav Freund, H Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. *Machine learning*, 28(2-3):133, 1997.

Kuan-Hao Huang and Hsuan-Tien Lin. A novel uncertainty sampling algorithm for cost-sensitive multiclass active learning. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 925–930. IEEE, 2016.

Daniel Kottke, Marek Herde, Christoph Sandrock, Denis Huseljic, Georg Krempl, and Bernhard Sick. Toward optimal probabilistic active learning using a bayesian approach. *Machine Learning*, 110(6):1199–1231, 2021.

Georg Krempl, Daniel Kottke, and Myra Spiliopoulou. Probabilistic active learning: A short proposition. In Torsten Schaub, Gerhard Friedrich, and Barry O'Sullivan, editors, *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI2014)*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 1049–1050, Prague, Czech Republic, August 2014. IOS Press. Short Paper.

David D Lewis and William A Gale. A sequential algorithm for training text classifiers. *arXiv preprint cmp-lg/9407020*, 1994.

Robert Munro. *Human-in-the-loop machine learning*. Manning Publications, New York, NY, July 2021.

Nicholas Roy and Andrew McCallum. Toward optimal active learning through monte carlo estimation of error reduction. *ICML, Williamstown*, 2:441–448, 2001.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, NJ, 3rd edition, 2009. ISBN 9780136042594.

Galuh Tunggadewi Sahid, Rahmad Mahendra, and Indra Budi. E-commerce merchant classification using website information. In *Proceedings of the 9th International Conference on Web Intelligence, Mining and Semantics*, pages 1–10, 2019.

H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294, 1992.

List of Figures

List of Tables

List of Abbreviations

A. Attachments

Table A.1: Category counts of usable given data sorted by originally given, originally given data that is English, originally given translated to English (+ English data), and supplementary data.

Category	Original	Original English	Translated	Additonal
Atm	8	2	4	4
Beauty	49	8	31	18
Bills And Household	19	4	9	10
Car	91	28	91	0
Children	9	0	4	5
Consumer Goods	36	5	28	7
Culture	16	1	10	6
Digital Services	20	12	16	4
Drugstore	8	2	3	5
Electronics	20	6	15	5
Fashion	28	6	22	6
Financial Services	10	0	4	6
Food And Drink	265	110	262	0
Freetime	16	2	8	8
Groceries	85	21	75	9
Health	73	14	64	8
House And Garden	47	11	44	3
Investments	7	1	2	5
Pets	26	6	18	8
Professional Services	37	15	32	5
Shopping Online	7	2	2	5
Sport	21	2	14	7
Travel	65	17	58	7

Table A.2: Variable importance, top 20 words from the vectorizer.

	Importance
hotel	0.091429
car	0.053077
hair	0.029760
auto	0.028876
station	0.017934
hairdresser	0.016447
flower	0.015114
spa	0.013156
parking	0.013083
stations	0.012124
barber	0.011965
internet	0.011843
services	0.011364
rental	0.010331
salon	0.009946
accommodation	0.009681
service	0.009430
bank	0.009072
cookies	0.007105
gas	0.006985