STUDY OF ACTIVE LEARNING ALGORITHMS

Internship Report Submitted By:

Kanupriya Jain
Institut Polytechnique de Paris

Under the guidance of:

Prof. Christophe Denis
LPSM, Sorbonne Université

Contents

Acknowledgement

I would like to express my sincere gratitude to everyone who contributed to my internship experience.

First, I extend my heartfelt thanks to ***Institut Polytechnique de Paris*** for providing me with this invaluable opportunity.

I am deeply grateful to my supervisor, ***Prof. Christophe Denis***, for his continuous guidance, support, and encouragement throughout the internship. His insights and expertise have been instrumental in my learning and professional growth.

Kanupriya

1 Introduction

Active learning is a crucial field within machine learning focused on enhancing the training process by carefully choosing the most informative data points for labeling. This report begins with an introduction to binary classification and the supervised learning framework. It then compares several passive learning classifiers with the Bayes classifier. The internship report explores various active learning algorithms, concentrating on three specific methods: Active Learning with Rejection, Uncertainty Sampling, and Query by Committee. We assess the performance of these algorithms using both synthetic data and real-world datasets for binary classification problems. Additionally, we aim to determine the theoretical bounds on the Empirical Risk in both active and passive learning frameworks.

2 Binary Classification

Definition It is a supervised learning algorithm that categorizes the new observations into one of two classes. We particularly focus on 0-1 classification where the output is either 1 or 0.

Examples :

- **Spam Detection:** Classifying emails as spam (1) or not spam (0).
- **Medical Diagnosis:** Determining if a patient has a certain disease (1) or not (0).
- **Sentiment Analysis:** Assessing whether a review is positive (1) or negative (0).

Types Of Classifers

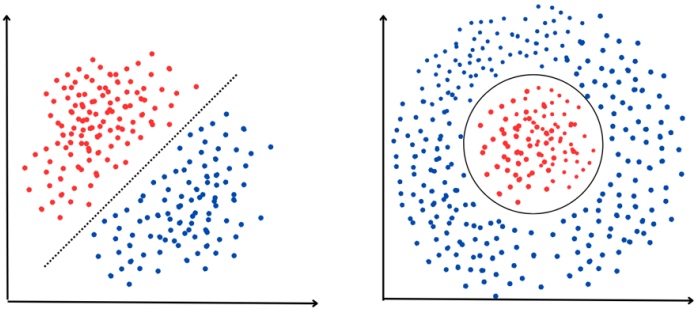
Classifiers can be largely separated into two categories, Linear and Non-linear classifiers. Linear Classification involves categorizing data points into discrete classes using a linear combination of explanatory variables. Conversely, Non-Linear Classification is used to distinguish instances that cannot be separated linearly.

Linear Classifiers:

- Logistic Regression
- Support Vector Machines (having *kernel*="linear")
- Stochastic Gradient Descent(SGD) Classifier

Non-Linear Classifiers :

- K-Nearest Neighbours
- Decision Tree Classification
- Random Forest



2.1 Supervised Classification Setting:

For our case of binary classification, consider the following notations-

Observation : We have our observation in the form of a tuple (X, Y) where $X \in \mathcal{X} = \mathbb{R}^d$ (Instance Space) and $Y \in \mathcal{Y} = \{0, 1\}$

Objective : Our objective is to find a classifier which is a measurable function $f : \mathbb{R}^d \to \{0, 1\}$

The mapping that needs to be learned in the framework used in supervised learning is parametrized by $\theta \in \Theta$. Our aim is to minimize the expected risk also called the *Misclassification Risk* i.e.

$$\min_{\theta \in \Theta} \mathcal{R}(f_\theta) \text{ where } \mathcal{R}(f_\theta) = \mathbb{E}[\ell(y, f_\theta(x)]$$

for some loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$.

The above expected risk is minimized for the *Bayes predictor* $f^* : \mathcal{X} \to \mathcal{Y}$ defined as

$$f^*(x') = \arg\min_{z \in \mathcal{Y}} \mathbb{E}[\ell(Y, z)|X = x']$$

The *Bayes risk* is the risk of all Bayes predictors:

$$\mathcal{R}^* = \mathbb{E}[\inf_{z \in \mathcal{Y}} \mathbb{E}[\ell(Y, z)|X]$$

The loss function that we consider is $\ell(y, z) = \mathbb{1}_{\{y \neq z\}}$. So, the misclassification risk is given by $R(f) = \mathbb{P}(f(X) \neq Y)$

The regression function is defined as $\eta(x) = \mathbb{P}(Y = 1|X = x)$

The Bayes classifier and Bayes risk in our case are given by

$$f^*(x) = \mathbb{1}_{\{\eta(x) \geq \frac{1}{2}\}} \text{ and } \mathcal{R}^* = \mathbb{E}[min\{\eta(X), 1 - \eta(X)\}]$$

.

We can observe that

$$\mathcal{R}^* = \mathcal{R}(f^*) = 1 - \mathbb{E}[g^*(X)]$$

where $g^*(.) = max\{\eta(.), 1 - \eta(.)\}$ is called the *score function.*

3 Emperical Risk Minimization

Given an i.i.d. sample $(X_1, Y_1), \ldots, (X_n, Y_n)$, our aim is to conduct supervised learning i.e. to find a map from *Instance Space* $\mathcal{X}$ to set of outputs $\mathcal{Y}$ called the *label space.* Here, we consider binary classification problem and take our $\mathcal{Y}$ to be $\{0, 1\}$.

3.1 Theoretical guarantees for passive framework

We want to find the theoretical bounds on the misclassification risk under the above-mentioned supervised learning setting under the parametric framework. So, we take a parameter class Θ. We also consider a parametric set of regression functions and classifiers as follows -

1. $\mathcal{F} = \{\eta_\theta : \theta \in \Theta\}$
2. $\mathcal{G}_\Theta = \{f_\theta : \theta \in \Theta\}$, set of classifiers of the form $f_\theta = \mathbb{1}_{\{\eta_\theta(x) \geq \frac{1}{2}\}}$.

The above expected risk 2.1 is approximated by the finite sum and the algorithm returns a prediction function $f_{\hat{\theta}_n}$ and the Emperical Risk Minimizer is defined as

$$\hat{\theta} \in \arg\min_{\theta \in \Theta} \hat{\mathcal{R}}_n(f_\theta) \quad \text{where } \hat{\mathcal{R}}_n(f_\theta) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{f_\theta(x_i) \neq y_i\}}$$

and its oracle counterpart is defined as

$$\theta^* \in \arg\min_{\theta \in \Theta} \mathcal{R}(f_\theta) \text{ where } \mathcal{R}(f_\theta) = \mathbb{E}[\mathbb{1}_{\{f_\theta(\mathcal{X}) \neq \mathcal{Y}\}}]$$

We would need the following definition to understand the theorem and its proof -

Definition (ϵ-cover)

Consider $\mathcal{F}$, a class of regression functions defined as $\eta : \mathbb{R}^d \to [0, 1]$. Let $\mathcal{F}_\epsilon = \{\eta^{(1)}, \ldots, \eta^{(N)}\}$ be a finite collection of functions $\mathbb{R}^d \to [0, 1]$ such that

$$\mathcal{F} \subset \bigcup_{\eta' \in \mathcal{F}_\epsilon} S_{\eta', \epsilon}$$

where $S_{\eta', \epsilon}$ is the ball of all functions $\psi : \mathbb{R}^d \to [0, 1]$ with

$$||\psi - \eta'||_\infty \leq \sup_x |\psi(x) - \eta'(x)| < \epsilon$$

$\mathcal{F}_\epsilon$ is called ϵ-*cover* of $\mathcal{F}$. The minimal value of $|\mathcal{F}_\epsilon|$ over all ϵ-covers is called the ϵ-*covering number* ($\mathcal{N}_\epsilon$).

We use the following theorem in the proof of theoretical guarantee in the case of passive framework-

Theorem 1. *For the error probability of the plug-in decision g of the form* $g_\theta = \mathbb{1}_{\{\eta_\theta(x) \geq \frac{1}{2}\}}$*, we have*

$$\mathbb{P}(g_\theta(X) \neq Y) - R(g_{\theta^*}(X)) = 2 \int_{\mathbb{R}^d} \left| \eta(x) - \frac{1}{2} \right| \mathbb{1}_{\{g_{\theta^*}(x) \neq g_{\theta^*}(x)\}} \mu(dx),$$

and

$$\mathbb{P}(g_\theta(X)) \neq Y) - R(g_{\theta^*}(X)) \leq 2 \int_{\mathbb{R}^d} |\eta_\theta(x) - \eta_{\theta^*}(x)| \mu(dx) = 2\mathbb{E}|\eta_\theta(X) - \eta_{\theta^*}(X)|.$$

where g_{θ^*} *is Baye's classifier and* $\mathcal{R}(g_\theta) = \mathbb{E}[\mathbb{1}_{\{g_\theta(X) \neq Y\}}]$.

Following is the result representing the performance of passive framework-

Theorem 2. *Let $(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$ be $i.i.d.$ data points. Let $\mathcal{F}$ be a parametric class of regression functions $\mathcal{F} = \{\eta_\theta : \theta \in \Theta\}$ and $\mathcal{G}_\Theta = \{f_\theta : \theta \in \Theta\}$ be set of classifiers of the form $f_\theta = \mathbb{1}_{\{\eta_\theta(x) \geq \frac{1}{2}\}}$ both parametrized by θ and Θ is the set of parameters. Let us also assume the following :*

1. *Θ, the class of regression function admits an ϵ-cover Θ_ϵ.*
2. *We assume that the Bayes predictor $f^* = f_{\theta^*} \in \mathcal{G}_\Theta$.*
3. *We also assume that the $\hat{\theta} \in \Theta_\epsilon$*
4. ***Lipschitz Condition***

   $$\text{For any } x,\ ||\eta_\theta(x) - \eta_{\theta'}(x)|| \leq ||\theta - \theta'||\ ||x||$$

5. ***Compactness***

   The set of instance space $\mathcal{X}$ is compact which implies there exists a closed ball centered at the origin with a finite radius R such that $\mathcal{X} \subset \overline{B}(0, R)$

   Then, the following inequality is satisfied

   $$\mathbb{E}[R(f_{\hat{\theta}}) - R(f^*)] \leq O\left(\sqrt{\frac{\log(2|\Theta_\epsilon|)}{n}} + \epsilon\right)$$

Remark:

- If $\Theta \subset \mathbb{R}^M$ and compact, then $|\Theta_\epsilon| \leq c\left(\frac{1}{\epsilon}\right)^M$
- The case discussed above is not used in practice due to our choice of non-convex loss function.

Proof:

$$\begin{aligned} R(f_{\hat{\theta}}) - R(f^*) &= R(f_{\hat{\theta}}) - \hat{R}(f_{\hat{\theta}}) + \hat{R}(f_{\hat{\theta}}) - R(f_{\theta^*}) \\ &\leq R(f_{\hat{\theta}}) - \hat{R}(f_{\hat{\theta}}) + \hat{R}(f_{\theta^*}) - R(f_{\theta^*}) \\ &\leq 2 \max_{\theta \in \Theta} |\hat{R}(f_\theta) - R(f_\theta)| \end{aligned} \tag{1}$$

Now, our aim is to find a bound on $\mathbb{E}[\max_{\theta \in \Theta} |\hat{R}(f_\theta) - R(f_\theta)|]$.

Case(I) $|\Theta| < \infty$

Let $|\Theta| = \alpha$. First, we will show that

$$\int_0^\infty \alpha \mathbb{P}(|\hat{R}(f_\theta) - R(f_\theta)| \geq t) dt \leq \int_0^\infty \min(\exp(\ln(2\alpha) - 2nt^2), \alpha) dt \tag{2}$$

Consider, $\hat{R}(f_\theta) = \frac{1}{n} \sum_{i=1}^n \underbrace{\mathbb{1}_{\{f_\theta(X_i) \neq Y_i\}}}_{\mathcal{Z}_i}$. We can observe that $\mathcal{Z}_i \sim Ber(R(f_\theta))$, $\mathcal{Z}_i$ are iid random variables and $0 \leq \mathcal{Z}_i \leq 1$. Now, by *Hoeffding's Inequality* we get,

$$\begin{aligned} \mathbb{P}(|\hat{R}(f_\theta) - R(f_\theta)| \geq t) &= \mathbb{P}(|\frac{1}{n} \sum_{i=1}^n \mathcal{Z}_i - R(f_\theta)| \geq t) \\ &= \mathbb{P}(|\sum_{i=1}^n \mathcal{Z}_i - n\mathbb{E}[\mathcal{Z}_1]| \geq nt) \\ &\leq 2 \exp\left(\frac{-2n^2t^2}{n}\right) \\ &= 2 \exp\left(-2nt^2\right) \end{aligned}$$

We also know that $\mathbb{P}(|\hat{R}(f_\theta) - R(f_\theta)| \geq t) \leq 1$. Therefore, we get

$$\int_0^\infty \alpha\mathbb{P}(|\hat{R}(f_\theta) - R(f_\theta)| \geq t)dt \leq \int_0^\infty \min(2\alpha\exp(-2nt^2), \alpha)dt \quad = \int_0^\infty \min(\exp(\ln(2\alpha) - 2nt^2), \alpha)dt$$

Now,

$$\begin{aligned}
\mathbb{E}[\max_{\theta\in\Theta} |\hat{R}(f_\theta) - R(f_\theta)|] &= \int_0^\infty \mathbb{P}(\max_{\theta\in\Theta} |\hat{R}(f_\theta) - R(f_\theta)| \geq t)dt \\
&\leq \int_0^\infty \mathbb{P}(\bigcup_{\theta\in\Theta} |\hat{R}(f_\theta) - R(f_\theta)| \geq t)dt \\
&\leq \int_0^\infty \sum_{\theta\in\Theta} \mathbb{P}(|\hat{R}(f_\theta) - R(f_\theta)| \geq t)dt \quad (Boole's Inequality) \\
&\leq \int_0^\infty \alpha\, \mathbb{P}(|\hat{R}(f_\theta) - R(f_\theta)| \geq t)dt \\
&\leq \int_0^\infty \min(\exp(\ln(2\alpha) - 2nt^2), \alpha)dt \quad (by\ (2)) \\
&= \int_0^{\sqrt{\frac{\ln(2\alpha)}{2n}}} \alpha dt + \int_{\sqrt{\frac{\ln(2\alpha)}{2n}}}^\infty \exp(\ln(2\alpha) - 2nt^2)dt \\
&= \alpha\sqrt{\frac{\ln(2\alpha)}{2n}} + \alpha\sqrt{\frac{\pi}{2n}} \left(\frac{2}{\sqrt{\pi}} \int_{\sqrt{\ln(2\alpha)}}^\infty \exp(-z^2)dz\right) \\
&= \sqrt{\frac{\ln(2\alpha)}{2n}} + \alpha\sqrt{\frac{\pi}{2n}} \quad erfc\left(\sqrt{\ln(2\alpha)}\right)
\end{aligned}$$

Case(II) $|\Theta| = \infty$ and Θ admits an ϵ-cover Θ_ϵ

Now, we define $\tilde{\theta} \in \arg\min_{\theta\in\Theta_\epsilon} \mathcal{R}(\theta)$

$$R(f_{\hat{\theta}}) - R(f^*) = R(f_{\hat{\theta}}) - R(f_{\tilde{\theta}}) + R(f_{\tilde{\theta}}) - R(f_{\theta^*}) \tag{3}$$

Since, Θ_ϵ admits an ϵ-cover, $\mathcal{F}$ also admits an ϵ-cover $\mathcal{F}_\epsilon$ by the properties of regression function. So, we choose θ_ϵ s.t. $||\eta_{\theta_\epsilon} - \eta_{\epsilon^*}|| \leq \epsilon$

Now, by the above theorem 1,

$$\begin{aligned}
R(f_{\theta_\epsilon}) - R(f_{\theta^*}) &\leq 2\mathbb{E}[|\eta_{\theta_\epsilon}(X) - \eta_{\epsilon^*}(X)|] \\
&\leq 2\epsilon
\end{aligned} \tag{4}$$

Also, by definition of $\tilde{\theta}$, $R(f_{\tilde{\theta}}) \leq R(f_{\theta'})$

$$R(f_{\tilde{\theta}}) - R(f_{\theta_\epsilon}) \leq 0 \quad \forall\theta' \in \Theta_\epsilon \tag{5}$$

Now, by (4) and (5) we can say that,

$$\begin{aligned}
R(f_{\hat{\theta}}) - R(f^*) &= R(f_{\hat{\theta}}) - R(f_{\tilde{\theta}}) + \underbrace{R(f_{\tilde{\theta}}) - R(f_{\theta_\epsilon})}_{\leq 0} + \underbrace{R(f_{\theta_\epsilon}) - R(f_{\theta^*})}_{2\epsilon} \\
&\leq R(f_{\hat{\theta}}) - R(f_{\tilde{\theta}}) + 2\epsilon
\end{aligned} \tag{6}$$

We know $|\Theta_\epsilon| < \infty$ and $\hat{\theta}, \tilde{\theta} \in \Theta_\epsilon$, so we can use Case(I) to bound $R(f_{\hat{\theta}}) - R(f_{\tilde{\theta}})$

Hence the proof is completed. □

Additional Result:

Theorem 3. *Consider a square $[0,1]^2$. The square is divided into smaller squares of side length r. Let C_i denote the i^{th} sub-square. We have labeled points inside the square and we are doing 0-1 classification on this square. Let η_{θ^*} denotes the regression function corresponding to the Bayes classifier on the entire square and $\eta_{\theta_i{}^*}$ denotes the regression function corresponding to the Bayes classifier on the sub square C_i. Now, if we define a new regression function $\tilde{\eta}(x) = \sum_{i=1}^{\frac{1}{r^2}} \eta_{\theta_i{}^*}(x) \mathbb{1}_{x \in C_i}$. We also assume the following-*

1. *Θ, the class of regression function admits an ϵ-cover Θ_ϵ.*
2. ***Lipschitz Condition***

$$\textit{For any } x, \ ||\eta_\theta(x) - \eta_{\theta'}(x)|| \leq ||\theta - \theta'||\,||x||$$

Then $\eta_{\theta^*} = \tilde{\eta}$

Proof: We will start with $\mathbb{E}[|\tilde{\eta}(X) - \eta_{\theta^*}(X)|]$ and we will bound this term in terms of r.

$$\begin{aligned}
\mathbb{E}[|\tilde{\eta}(X) - \eta_{\theta^*}(X)|] &= \mathbb{E}[|\sum_{i=1}^{\frac{1}{r^2}} \eta_{\theta_i{}^*}(X)\mathbb{1}_{X \in C_i} - \eta_{\theta^*}(X)|] \\
&\leq \sum_{i=1}^{\frac{1}{r^2}} \mathbb{E}[|\eta_{\theta_i{}^*}(X) - \eta_{\theta^*}(X)|\mathbb{1}_{X \in C_i}] \\
&= \sum_{i=1}^{\frac{1}{r^2}} \mathbb{E}[|\eta_{\theta_i{}^*}(X) - \eta_{\theta^*}(X)|\,|\mathbb{1}_{X \in C_i}] \ \ \mathbb{P}(X \in C_i) \\
&= \sum_{i=1}^{\frac{1}{r^2}} \mathbb{E}[|\eta_{\theta_i{}^*}(X) - \eta_{\theta^*}(X)|\,|\mathbb{1}_{X \in C_i}] \ \ r^2
\end{aligned}$$

Let x_i be a point such that $x_i \in C_i$. Then,

$$\begin{aligned}
\mathbb{E}[|\eta_{\theta_i{}^*}(X) - \eta_{\theta^*}(X)|\,|\mathbb{1}_{X \in C_i}] &= \mathbb{E}[|\eta_{\theta_i{}^*}(X) - \eta_{\theta_i^*}(x_i)|\,|\mathbb{1}_{X \in C_i}] + \mathbb{E}[|\eta_{\theta_i{}^*}(x_i) - \eta_{\theta^*}(x_i)|\,|\mathbb{1}_{X \in C_i}] \\
&\quad + \mathbb{E}[|\eta_{\theta^*}(x_i) - \eta_{\theta^*}(X)|\,|\mathbb{1}_{X \in C_i}]
\end{aligned} \tag{7}$$

By the Lipschitz condition on $\eta_{\theta_i^*}$ and η_{θ^*} with respect to x, we can say the following-

$$\begin{aligned}
|\eta_{\theta_i{}^*}(X) - \eta_{\theta_i^*}(x_i)| &\leq L_1||X - x_i|| \\
&\leq L_1\sqrt{2}r
\end{aligned} \tag{8}$$

$$\begin{aligned}
|\eta_{\theta^*}(X) - \eta_{\theta^*}(x_i)| &\leq L_2||X - x_i|| \\
&\leq L_2\sqrt{2}r
\end{aligned} \tag{9}$$

By the Lipschitz condition on $\theta \in \Theta$, we have

$$
\begin{aligned}
|\eta_{\theta_i{}^*}(x_i) - \eta_{\theta^*}(x_i)| &\leq ||\theta_i^* - \theta^*||\ \ ||x_i|| \\
&\leq \sqrt{2}||\theta_i^* - \theta^*||
\end{aligned} \tag{10}
$$

Now, we have to bound the term $|\eta_{\theta_i{}^*}(x_i) - \eta_{\theta^*}(x_i)|$. We know Θ admits ϵ-cover which implies F, the class of regression functions. So, for $\epsilon = r$ we choose ${\theta_\epsilon}^1$ and ${\theta_\epsilon}^2$ in the ϵ-cover Θ_ϵ such that

$$
||\theta_i^* - {\theta_\epsilon}^1|| \leq r \quad \text{and} \quad ||\theta^* - {\theta_\epsilon}^2|| \leq r
$$

We can also note that ${\theta_\epsilon}^1$ and ${\theta_\epsilon}^2$ are in Θ_ϵ, so $||{\theta_\epsilon}^1 - {\theta_\epsilon}^2||$ is of order $\epsilon = r$. Therefore,

$$
\begin{aligned}
||\theta_i^* - \theta^*|| &\leq ||\theta_i^* - {\theta_\epsilon}^1|| + ||{\theta_\epsilon}^1 - {\theta_\epsilon}^2|| + ||\theta^* - {\theta_\epsilon}^2|| \\
&\leq r + C'r + r \\
&\leq Cr
\end{aligned} \tag{11}
$$

Now, by (10) and (11), we can say that

$$
\begin{aligned}
|\eta_{\theta_i{}^*}(x_i) - \eta_{\theta^*}(x_i)| &\leq \sqrt{2}||\theta_i^* - \theta^*|| \\
&\leq \sqrt{2}\, Cr
\end{aligned} \tag{12}
$$

By, (7), (8), (9) and (13), we get the following

$$
\begin{aligned}
\mathbb{E}[|\eta_{\theta_i{}^*}(X) - \eta_{\theta^*}(X)|\,|\mathbb{1}_{X \in C_i}] &\leq L_1\sqrt{2}r + L_1\sqrt{2}r + \sqrt{2}\, Cr \\
&\leq C_2 r
\end{aligned} \tag{13}
$$

where $C_2 = L_1\sqrt{2} + L_1\sqrt{2} + C\sqrt{2}$

Now, we will bound the first expectation in terms of r to get the result.

$$
\begin{aligned}
\mathbb{E}[|\tilde{\eta}(X) - \eta_{\theta^*}(X)|] &\leq \sum_{i=1}^{\frac{1}{r^2}} \mathbb{E}[|\eta_{\theta_i{}^*}(X) - \eta_{\theta^*}(X)|\,|\mathbb{1}_{X \in C_i}]\ \ r^2 \\
&\leq \sum_{i=1}^{\frac{1}{r^2}} (C_2 r)\, r^2 \\
&= C_2 r
\end{aligned}
$$

As, we reduce the size of subsquares i.e. as $r \to 0$, the above expectation goes to 0 and we get the result. □

Additional Result 2:

Theorem 4. *Consider a square $[0,1]^2$. The square is divided into smaller squares of side length r. Let C_i denote the i^{th} sub-square. We have labeled points inside the square and we are doing 0-1 classification on this square. Let η_{θ^*} denotes the regression function corresponding to the Bayes classifier on the entire square and $\eta_{\theta_i{}^*}$ denotes the regression function corresponding to the Bayes classifier on the sub-square C_i. Now, if we define a new classifier as follows-*

$$\tilde{R}(x) = \sum_{i=1}^{\frac{1}{r^2}} R(g_{\hat{\theta}_i}(x)) \mathbb{1}_{x \in C_i}$$

where $\hat{\theta}_i \in \arg\min_\theta \frac{1}{N_i} \sum_{j=1}^{N_i} \mathbb{1}_{\{g_\theta(x_j) \neq y_j\}}$ and N_i = number of points in the square i.

$R^*(x)$ is the Bayes classifier and it can also be written as the following -

$$R^*(x) = \sum_{i=1}^{\frac{1}{r^2}} R(g_{\theta_i{}^*}(x)) \mathbb{1}_{x \in C_i}$$

where $\theta_i{}^*$ is the parameter associated to Bayes classifier of each square.

We also assume the following-

1. *Θ_i, the class of regression function admits an ϵ-cover ${\Theta_\epsilon}^i$ for each i^{th} square.*
2. *We also assume that value of regression function stays constant in each square.*
3. ***Lipschitz Condition***

$$\text{For any } x, \ ||\eta_\theta(x) - \eta_{\theta'}(x)|| \leq ||\theta - \theta'||\, ||x||$$

Then $\mathbb{E}[|\tilde{R}(X) - R^*(X)|] \to 0$ as $r \to 0$

Proof:

$$\begin{aligned}
\mathbb{E}[|\tilde{R}(X) - R^*(X)|] &= \mathbb{E}[|\sum_{i=1}^{\frac{1}{r^2}} R(g_{\hat{\theta}_i}(X)) \mathbb{1}_{x \in C_i} - \sum_{i=1}^{\frac{1}{r^2}} R(g_{\theta_i{}^*}(X)) \mathbb{1}_{x \in C_i}|] \\
&\leq \sum_{i=1}^{\frac{1}{r^2}} \mathbb{E}[|R(g_{\hat{\theta}_i}(X)) - R(g_{\theta_i{}^*}(X))| \mathbb{1}_{X \in C_i}] \\
&= \sum_{i=1}^{\frac{1}{r^2}} \mathbb{E}[|R(g_{\hat{\theta}_i}(X)) - R(g_{\theta_i{}^*}(X))| \,|\, \mathbb{1}_{X \in C_i}] \ \ \mathbb{P}(X \in C_i) \\
&= \sum_{i=1}^{\frac{1}{r^2}} \mathbb{E}[|R(g_{\hat{\theta}_i}(X)) - R(g_{\theta_i{}^*}(X))| \,|\, \mathbb{1}_{X \in C_i}] \ \ r^2 \\
&\leq \sum_{i=1}^{\frac{1}{r^2}} 2\, \mathbb{E}[|\eta_{\hat{\theta}_i}(X) - \eta_{\theta_i{}^*}(X)| \,|\, \mathbb{1}_{X \in C_i}] \ \ r^2 \qquad \{\text{ by (1)}\}
\end{aligned} \tag{14}$$

Now, We proceed with $\mathbb{E}[|\tilde{\eta}(X) - \eta_{\theta^*}(X)|]$ and we will bound this term in terms of r.

Let x_i be a point such that $X_i \in C_i$. Then,

$$\begin{aligned}\mathbb{E}[|\eta_{\hat{\theta}_i}(X) - \eta_{\theta_i{}^*}(X)|\,|\mathbb{1}_{X\in C_i}] &= \mathbb{E}[|\eta_{\hat{\theta}_i}(X) - \eta_{\hat{\theta}_i}(X_i)|\,|\mathbb{1}_{X\in C_i}] + \mathbb{E}[|\eta_{\hat{\theta}_i}(X_i) - \eta_{\theta_i{}^*}(X_i)|\,|\mathbb{1}_{X\in C_i}] \\ &\quad + \mathbb{E}[|\eta_{\theta_i{}^*}(X_i) - \eta_{\theta_i{}^*}(X)|\,|\mathbb{1}_{X\in C_i}]\end{aligned} \tag{15}$$

By the Lipschitz condition on $\eta_{\theta_i^*}$ with respect to x, we can say the following-

$$\begin{aligned}|\eta_{\theta_i{}^*}(X_i) - \eta_{\theta_i^*}(X)| &\le L_1||X - X_i|| \\ &\le L_1\sqrt{2}r\end{aligned} \tag{16}$$

By the Lipschitz condition on $\theta \in \Theta_i$, we have

$$\begin{aligned}|\eta_{\hat{\theta}_i}(X_i) - \eta_{\theta_i{}^*}(X_i)| &\le ||\hat{\theta}_i - \theta_i{}^*||\ \ ||X_i|| \\ &\le \sqrt{2}||\hat{\theta}_i - \theta_i{}^*||\end{aligned} \tag{17}$$

Now, we have to bound the term $|\eta_{\hat{\theta}_i}(X_i) - \eta_{\theta_i{}^*}(X_i)|$. We know Θ_i admits ϵ-cover which implies $\mathcal{F}$, the class of regression functions. So, for $\epsilon = r$ we choose $\theta_\epsilon{}^1$ and $\theta_\epsilon{}^2$ in the ϵ-cover $\Theta_\epsilon{}^i$ such that

$$||\hat{\theta}_i - \theta_\epsilon{}^1|| \le r \quad \text{and} \quad ||\theta_i{}^* - \theta_\epsilon{}^2|| \le r$$

We can also note that $\theta_\epsilon{}^1$ and $\theta_\epsilon{}^2$ are in $\Theta_\epsilon{}^i$, so $||\theta_\epsilon{}^1 - \theta_\epsilon{}^2||$ is of order $\epsilon = r$. Therefore,

$$\begin{aligned}||\hat{\theta}_i - \theta_i{}^*|| &\le ||\hat{\theta}_i - \theta_\epsilon{}^1|| + ||\theta_\epsilon{}^1 - \theta_\epsilon{}^2|| + ||\theta_\epsilon{}^2 - \theta_i{}^*|| \\ &\le r + C'r + r \\ &\le Cr\end{aligned} \tag{18}$$

Now, by (17) and (18), we can say that

$$\begin{aligned}|\eta_{\hat{\theta}_i}(X_i) - \eta_{\theta_i{}^*}(X_i)| &\le \sqrt{2}||\hat{\theta}_i - \theta_i{}^*|| \\ &\le \sqrt{2}\,Cr\end{aligned} \tag{19}$$

We also note that the value of regression function stays constant for a square. So we can say that-

$$|\eta_{\hat{\theta}_i}(X) - \eta_{\hat{\theta}_i}(X_i)| = 0 \tag{20}$$

By, (15), (16), (19) and (20), we get the following

$$\begin{aligned}\mathbb{E}[|\eta_{\theta_i{}^*}(X) - \eta_{\theta^*}(X)|\,|\mathbb{1}_{X\in C_i}] &\le L_1\sqrt{2}r + \sqrt{2}\,Cr \\ &\le C_2 r\end{aligned} \tag{21}$$

where $C_2 = L_1\sqrt{2} + C\sqrt{2}$

Now, we will bound the first expectation in terms of r to get the result.

$$\begin{aligned}\mathbb{E}[|\tilde{R}(X) - R^*(X)|] &\le \sum_{i=1}^{\frac{1}{r^2}} 2\,\mathbb{E}[|\eta_{\hat{\theta}_i}(X) - \eta_{\theta_i{}^*}(X)|\,|\mathbb{1}_{X\in C_i}]\ \ r^2 \\ &\le \sum_{i=1}^{\frac{1}{r^2}} 2\ (C_2 r)\, r^2 \\ &= 2\,C_2 r\end{aligned}$$

As, we reduce the size of subsquares i.e. as $r \to 0$, the above expectation goes to 0 and we get the result. $\square$

4 Comparison Of Passive classifiers with Bayes Classifier

We implement different methods of classification and calculate misclassification errors in each case to compare it with the error that we would get with the Bayes classifier i.e. $f^*(x) = \mathbb{1}_{\{\eta(x) \geq \frac{1}{2}\}}$.

4.1 $Y \mid X \sim$ **Ber**$(\eta(X))$

In this framework, for each $A \subset \mathcal{X}$, and $M \geq 1$, we sample i.i.d random variables $(X_i, Y_i)_{1 \leq i \leq M}$ such that -

1. $\forall i$, X_i is distributed according to $\Pi(\cdot \mid A)$
2. conditionally on X_i, Y_i is distributed according to Bernoulli distribution with parameter $\eta(X_i)$

For simplicity, we take $\mathcal{X} = \mathbb{R}^2$. We take our regression function as the following -

$$\eta(x) = \frac{1}{1 + \exp\left(f(x)\right)} \text{ where } f(x_1, x_2) = -3x_1 + 2x_2 - 1$$

The decision boundary will be the collection of points where $\eta(x) = \frac{1}{2}$ i.e. $f(x) = 0$. We generated 1000 points and our dataset along with the decision boundary looks like the following -

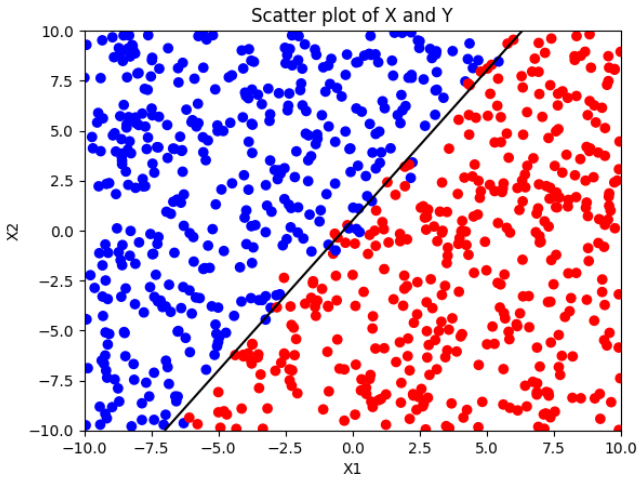


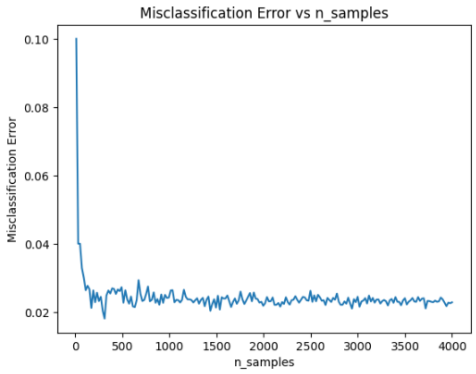
Figure 1

The red and blue dots represent the {0,1} labels corresponding to points in $\mathcal{X}$.

We implement Logistic Regression, K Nearest Neighbours, and Random Forest Algorithm along with the Bayes classifier. In each case, we generate 1000 data points 100 times as described in the framework mentioned above. Each time, we perform train and test split, training the model, and predictions. We then calculate the misclassification risk in each of the 100 iterations and take its mean at the end. In each case, we try to see the impact of varying the training size and n_sample (number of points generated in each iteration) on this mean misclassification error.

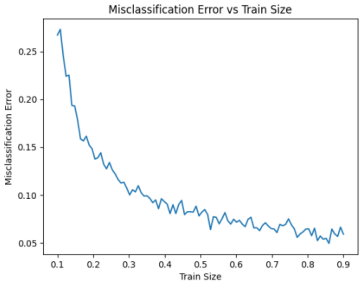
Logistic Regression

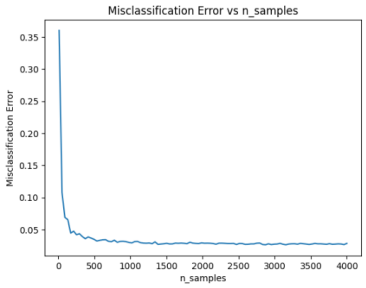
The mean misclassification error that we got in this case is **0.0239** . The result of varying the training size and n_sample (number of points generated in each iteration) on the mean misclassification error is the following -

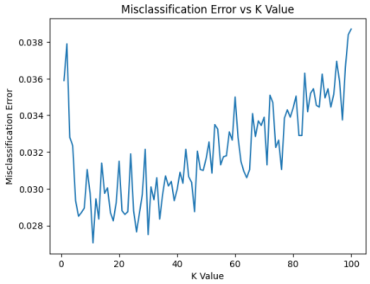


K Nearest Neighbours

The mean misclassification error obtained in this case is **0.0289** (with k =5). In this case, we also try to observe how the misclassification error varies with 'k'. The result of varying the training size and n_sample (number of points generated in each iteration) on the mean misclassification error is the following -



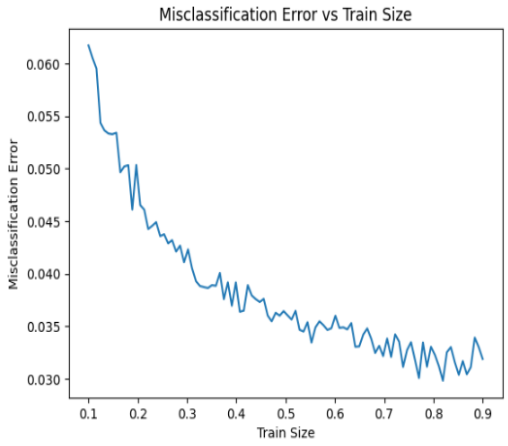


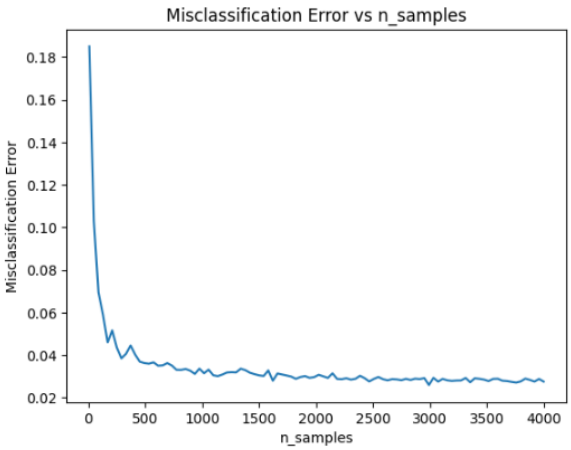


We observe that for small k, the model is highly sensitive to noise in the training data (high variance, low bias), which can lead to high misclassification errors. As k increases, the model smooths out the noise, leading to a decrease in the misclassification error. However, after a certain point, further increasing k might have led to the inclusion of points from other classes in the decision boundary (high bias, low variance), causing the misclassification error to increase again.

Random Forest

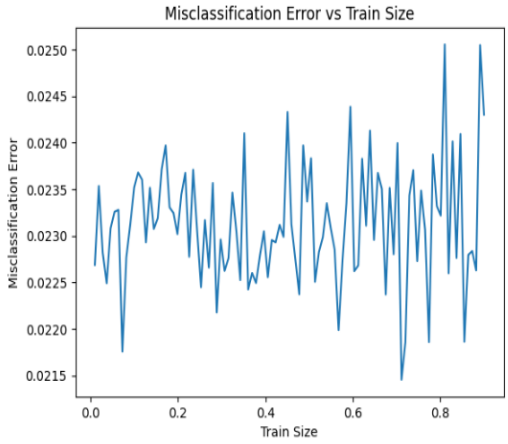
The average misclassification error we obtained in this case was **0.031**. We use `n_estimators` $= 50$ in each instance. The effect of varying the training size and n_sample (the number of points generated in each iteration) on the average misclassification error is as follows -

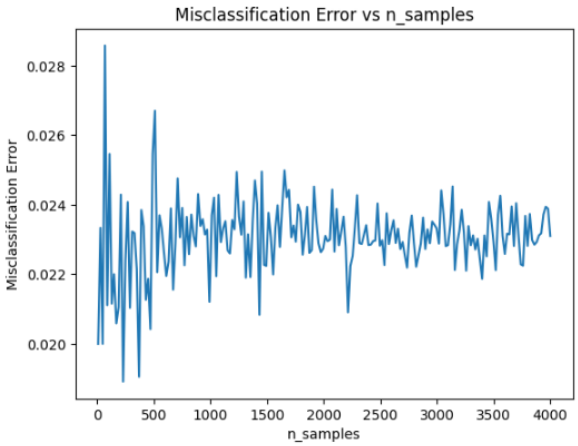




Bayes Classifier

The average misclassification error we obtained in this scenario was **0.0234**. We can observe that misclassification in this case is the least as expected. Training is not necessary here. Instead, we directly predict on the test set in each iteration to compute the error. The effect of changing the training size and n_sample (the number of points generated in each iteration) on the average misclassification error is as follows -





4.2 $Y \sim$ **Ber**(p)**and**$X \mid Y \sim \mathcal{N}_{\mathbb{R}^d}(\mu_Y, Id)$

In this framework, for each $A \subset \mathcal{X}$, and $M \geq 1$, we will sample i.i.d random variables $(X_i, Y_i)_{1 \leq i \leq M}$ such that -

1. $\forall i$, Y_i is distributed according to Bernoulli distribution with parameter p
2. conditionally on Y_i, X_i is distributed according to Normal distribution with parameter (μ_Y, Id)

For a general multivariate Gaussian distribution with mean μ and the covariance matrix Σ, the density is given by -

$$f(x) = \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} e^{\frac{-1}{2}(x-m)^T \Sigma^{-1}(x-m)}$$

Now, for a two-class problem, the random variable X has a density $pf_1(x) + (1-p)f_0(x)$ where $X|Y = 1 \sim f_1, X|Y = 0 \sim f_0$, f_1 and f_0 are both multivariate normal with parameters m_i and Σ_i, $i = 0$ and 1. In this case,

$$\eta(x) = \mathbb{P}(Y = 1|X) = \frac{pf_1(x)}{pf_1(x) + (1-p)f_0(x)}$$

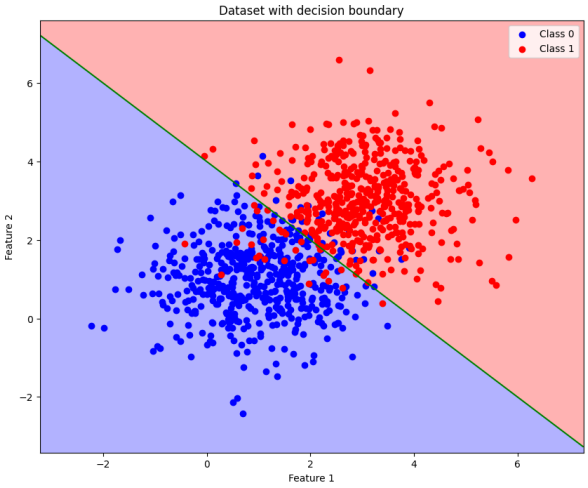
and Bayes classifier is given by -

$$f^*(x) = \begin{cases} 1 & \text{if } \eta(x) \geq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases} = \begin{cases} 1 & \text{if } pf_1(x) \geq (1-p)f_0(x) \\ 0 & \text{otherwise} \end{cases}$$

The decision boundary, in this case, will be the set of points where $\eta(x) = \frac{1}{2}$. The dataset consisting of 1000 points and the corresponding decision boundary is illustrated as follows:

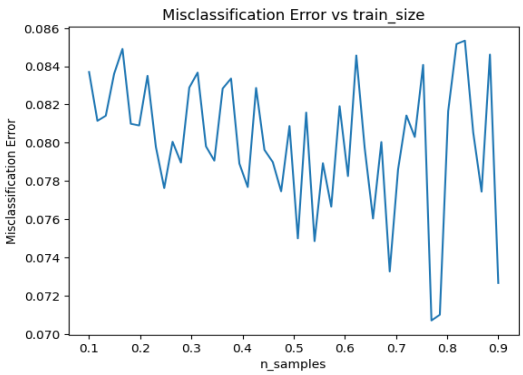
In this scenario, we will assess the effectiveness of Logistic Regression, K Nearest Neighbours, Random Forest Algorithm, and the Bayes classifier. We will apply the same procedures outlined in section 4.1. Specifically, we will generate 1000 data points 100 times as per the aforementioned framework.

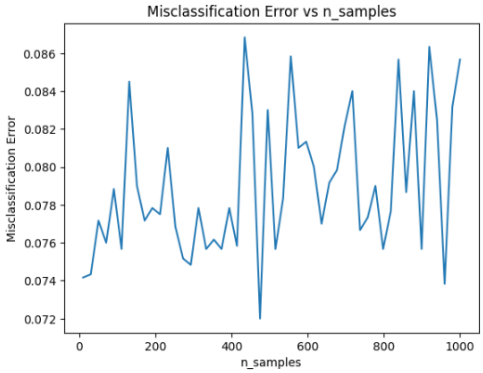
For each method, we will repeat the process of splitting the data into training and testing sets, training the model, making predictions, and then calculating the misclassification risk across 100 iterations. The average misclassification error across these iterations will be computed. Throughout this analysis, we aim to understand how varying the training size and the number of data points generated per iteration (n_sample) affects this average misclassification error.



Logistic Regression

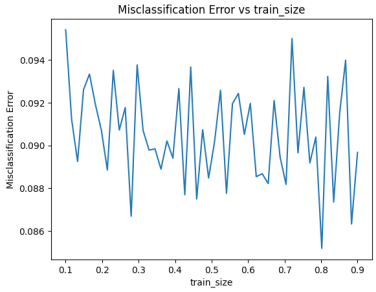
The mean misclassification error that we got in this case was **0.0783** and the standard deviation is **0.01877571569874235**. The result of varying the training size and n_sample (number of points generated in each iteration) on the mean misclassification error is the following -

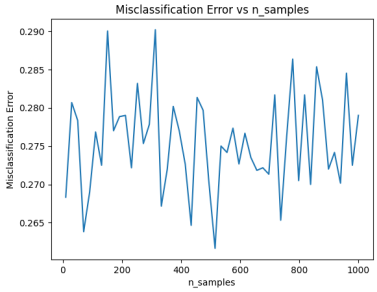




K Nearest Neighbours

The average misclassification error obtained was **0.0886**, with a standard deviation of **0.01679404656418458**. Once more, we examined how the misclassification error changes with different values of 'k'. The impact of varying the training size and n_sample (number of points generated in each iteration) on the mean misclassification error is as follows:

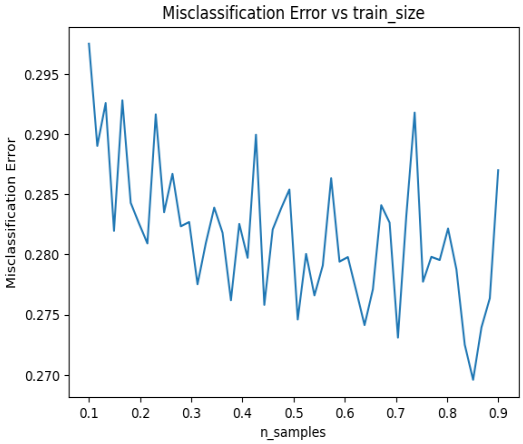


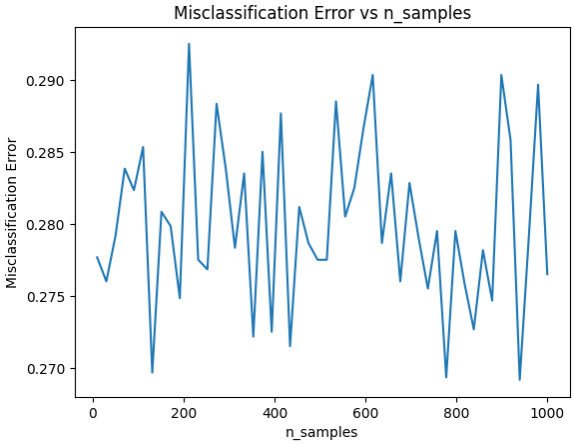


We observe that misclassification error decreases as k increases which is unlike the previous case. This might be due to the fact that dataset might have less noise or be perfectly balanced in a way that more neighbors always contribute to a more accurate prediction.

Random Forest

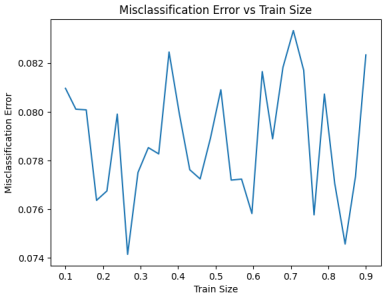
The average misclassification error we obtained in this case was **0.0891** and the standard deviation is **0.0223313680727357l4**. We used `n_estimators` $= 50$ in each instance. The effect of varying the training size and n_sample (the number of points generated in each iteration) on the average misclassification error is as follows -

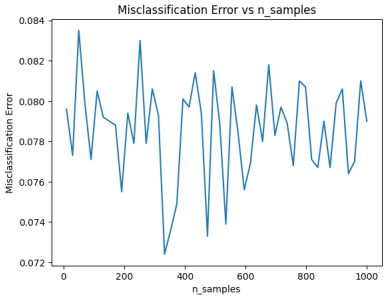




Bayes Classifier

In this particular case, we achieved an average misclassification error of **0.0766**, which is the lowest among all methods as anticipated. Here, training was unnecessary, and we directly performed predictions on the test set using the Bayes classifier in each iteration to calculate the error. The impact of varying the training size and n_sample (the number of points generated in each iteration) on the average misclassification error is as follows:





General observations for both datasets:

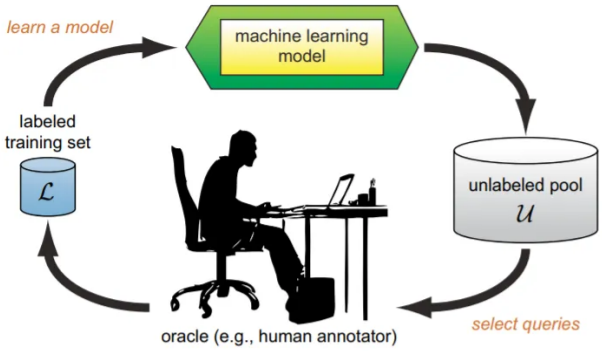
1. All models show improvement in misclassification error with an increase in training size and the number of samples per iteration.
2. The Random Forest model seems to have a highest and Bayes classifier have the least baseline error compared to other models.
3. KNN but also shows a consistent decrease in error with increasing data. The choice of parameters (such as k for KNN) plays a significant role in the model's performance, as evidenced by the fluctuations in the misclassification error with varying k.

These observations highlight the importance of parameter tuning and sufficient data in training models to achieve better accuracy and lower misclassification errors.

5 Active Learning Algorithm

Definition: Active Learning aims to reduce the number of labeled data required for training and selecting the data that needs to be labeled.

Active Learning Framework :



Motivation :

- Acquiring labeled data can be costly and time-consuming.
- In many domains, such as medical imaging or legal document analysis, expert labeling is required, increasing costs.

Sampling Strategies:

Active learning techniques choose the most informative and valuable data to improve the learning process. In this process, three main strategies are the following -

1. **Stream based Sampling:** An active learning technique for training models on continuous data streams, where each sample is sequentially evaluated for labeling.
2. **Pool based Sampling:** It begins with a large pool of unlabeled data and then ranks all samples in the pool and then selects the best ones to query.
3. **Membership Query Synthesis:** In this scenario, the active learning algorithm generates new, unlabeled instances within the input space and queries an oracle for their labels.

Query Strategies:

One of the key principles of active learning is to identify at each step the region of the instance space where the label requests should be made, called *Uncertain Region*, also known as *Disagreement Region*. To identify this region, we discuss here three main strategies which are -

1. Uncertainty Sampling
2. Query By Committee (QBC)
3. Rejection

5.1 Introduction to Active Learning with Rejection Method

We use an active learning approach utilizing the concept of *Rejection*. Traditionally, in this learning method instead of simply assigning a class label to every instance it encounters, the model may choose to "reject" predicting a label for instances where its confidence falls below a certain threshold. In rejection, the focus on the uncertain region arises after designing a learning model, where a test point is rejected to prevent a potential misprediction. This is particularly valuable in applications like medical diagnosis, where an incorrect prediction can have serious consequences.

On the other hand, in active learning, the uncertain region is utilized during the training phase to enhance the model's performance gradually. This is achieved by selectively requesting labels for instances where classification is challenging.

In our algorithm, we employ rejection during each step k of the training process to estimate the uncertain region $A_k \subset \mathcal{X}$ using accumulated information up to that step. Subsequently, we sample points from region A_k and request their labels. Using these labeled examples, we obtain an estimator $\hat{f}_k$, which is used to evaluate the confidence in predictions for each $x \in A_k$. Points with low confidence are rejected, defining the next uncertain region A_{k+1} . This iterative process progressively reduces the portion of the instance space $\mathcal{X}$ where a model needs to be constructed.

5.2 Description Of the Algorithm:

- We take our initial uncertain region as $A_0 = \mathcal{X} = [0,1]^d$. We will use the score function as $g(x) = max\{\eta(x), 1-\eta(x)\}$
- We also take fixed number of label requests N (called the budget).
- We define a sequence of positive numbers $(\epsilon_k)_{k\geq 0}$ as a *sequence of rejection rates* and another sequence $(N_k)_{k\geq 0}$ defined as $N_0 = \lfloor\sqrt{N}\rfloor$ and $N_{k+1} = \lfloor c_N N_k\rfloor$ with $c_N > 1$.
- We will construct a sequence of uncertain regions $(A_k)_{k\geq 1}$ and $\hat{\eta}_k$ on A_k.

5.2.1 Initialization Phase

First, our algorithm performs an initialization phase:

- Initially, the learner requests the labels Y of N_0 points $X_1, \ldots, X_{N_0}$ sampled in A_0 according to $\Pi_0 = \Pi$.
- Based on the initial labeled data $D_{N_0} = \{(X_1, Y_1), \ldots, (X_{N_0}, Y_{N_0})\}$, an estimator $\hat{\eta}_0$ of η on A_0 is computed, and an initial classifier $\hat{f}_0 = \mathbb{1}_{\{\hat{\eta}_0 \geq \frac{1}{2}\}}$ is provided.
- An estimator of the score function $\hat{g}_0(x) = \max(\hat{\eta}_0(x), 1-\hat{\eta}_0(x))$ associated with $\hat{\eta}_0$ is computed.

5.2.2 Iteration Phase

Afterwards, our algorithm iterates over a finite number of steps until the label budget N has been reached. Step $k \geq 1$ is described below.

1. **Threshold Calculation**:
   - Based on the previous uncertain region A_{k-1}, a constant λ_k is computed. This constant is defined by the sequence of rejection rates $(\epsilon_k)_{k\geq 0}$ explicitly.
2. **Uncertain Region Construction**:
   - This constant λ_k is used to construct the current uncertain region A_k, which is the set where the previous classifier $\hat{f}_{k-1}(x) = \mathbb{1}_{\{\hat{\eta}_{k-1}(x) \geq \frac{1}{2}\}}$ might fail and thus abstains from labeling:

     $$\{x \in A_{k-1} : \hat{g}_{k-1}(x) \leq \lambda_k\}$$

     where $\hat{g}_{k-1}(x) = \max(\hat{\eta}_{k-1}(x), 1-\hat{\eta}_{k-1}(x))$.
3. **Data Sampling and Model Update**:
   - According to $\pi(\cdot|A_k)$, the learner samples i.i.d. (X_i, Y_i), $i = 1, \ldots, \lfloor N_k \epsilon_k \rfloor$, used to compute an estimator $\hat{\eta}_k$ of η on A_k and $\hat{\eta}$ is defined as the following

     $$\hat{\eta} = \sum_{j=0}^{k-1} \hat{\eta}_j \mathbb{1}_{\{A_j \setminus A_{j+1}\}} + \hat{\eta}_k \mathbb{1}_{A_k}$$

4. **Classifier Update**:
   - The learner updates the classifier over the whole space X as follows: after the iteration process, the resulting active classifier with rejection is defined pointwise as

$$\hat{f}(x) = \mathbb{1}_{\{\hat{\eta}(x) \geq \frac{1}{2}\}}.$$

5.3 Theoretical result in the case of non-parametric setting

We present here the theoretical bounds on the risk in the case of active framework. Before proceeding further, we need the following concepts -

Definition (Histogram Rule)

Let us denote by $\mathcal{C}_r = \{R_i, i = 1, \ldots, r^{-d}\}$ a cubic partition of $[0,1]^d$ with edge length $r > 0$.

Consider a labeled sample $D_{N_A} = \left\{(X_A^1, Y_1), \ldots, (X_A^{N_A}, Y_{N_A})\right\}$ of size $N_A \geq 1$, such that X_A^i $(i = 1, \ldots, N_A)$ is distributed according to $\Pi(\cdot|A)$. The histogram rule on A is defined as follows. Let $R_i \in \mathcal{C}_r$ with $R_i \cap A \neq \emptyset$. For all $x \in R_i$,

$$\hat{\eta}_{A,N_A,r}(x) = \frac{\Pi(A)}{\Pi(R_i|A)} \frac{1}{N_A} \sum_{j=1}^{N_A} Y_j \mathbb{1}_{\{X_j \in R_i\}}.$$

Case of Active Learning framework:

Assumptions:

We assume our instance space to be $\mathcal{X} = [0,1]^d$ and the following assumptions to study the rates of convergence-

1. **(Smoothness assumption)** The regression function η is β-Hölder for some $\beta \in (0,1)$, that is, there exists $s > 0$, such that for all $x, z \in [0,1]^d$

$$|\eta(x) - \eta(z)| \leq s||x - z||_\infty^\beta$$

2. **(Strong density assumption)** The marginal probability admits a density p_X and there exist constants $\mu_{min}, \mu_{max} > 0$ such that for all $x \in [0,1]^d$ with $p_X(x) > 0$, we have

$$\mu_{min} \leq p_X(x) \leq \mu_{max}$$

3. **(Score regularity assumption)** Let $g(x) = max\{\eta(x), 1 - \eta(x)$ bethe score function. The random variable $g(X)$ admits a bounded density (bounded by $C > 0$).

Here is the theoretical result highlighting the performance of the active learning algorithm-

Theorem 5. *Let N be the label budget, and $\delta \in \left(0, \frac{1}{2}\right)$. Let the above three assumptions are fulfilled. At each step $k \geq 0$ of the algorithm presented in Sect. 5.2, we consider*

(i) $\hat{\eta}_k := \hat{\eta}_{A_k, \lfloor N_k \epsilon_k \rfloor, r_k}$, *with* $r_k = N_k^{-1/(2\beta+d)}$,

(ii) and define $(\epsilon_k)_{k \geq 0}$ *as* $\epsilon_0 = 1$, *and for* $k \geq 1$,

$$\epsilon_k = \min\left(1, \log\left(\frac{N}{\delta}\right) \log(N) N_{k-1}^{-\beta/(2\beta+d)}\right).$$

Then with probability at least $1 - \delta$, *the resulting classifier defined in Equation (3.3) satisfies*

$$R(f_{\hat{\eta}}) - R(f^*) \leq \tilde{O}\left(N^{-\frac{2\beta}{2\beta+d}}\right),$$

where $\tilde{O}$ *hides some constants and logarithmic factors.*

5.4 Theoretical result in the case of parametric setting

Theorem 6. *Consider a square $[0,1]^2$. The square is divided into smaller squares of side length r. Let R_i denote the i^{th} sub-square. We have labeled points inside the square and are doing 0-1 classification in the active learning framework on this square, building uncertainty regions denoted by A_k. Let η_{θ^*} denote the regression function corresponding to the Bayes classifier on the entire square, and $\eta_{\theta_k^*}$ denote the regression function corresponding to the Bayes classifier on the uncertainty region A_k.*

Let L be defined as

$$L = \max\left\{ j \geq 1 : N > \sum_{k=0}^{j} \lfloor N_k \epsilon_k \rfloor \right\}.$$

Now, define a new regression function $\tilde{\eta}(x) = \sum_{k=1}^{L} \eta_{\theta_k^*}(x) \mathbb{1}_{x \in A_k}$. We also assume the following:

1. $A_k = \bigcup_{R_i \cap A_k \neq \emptyset} R_i$
2. *Θ, the class of regression functions, admits an ϵ-cover Θ_ϵ.*
3. ***Lipschitz Condition:***

$$\textit{For any } x, \quad \|\eta_\theta(x) - \eta_{\theta'}(x)\| \leq \|\theta - \theta'\| \cdot \|x\|,$$

and

$$\textit{For any } \theta, \quad \|\eta_\theta(x_1) - \eta_\theta(x_2)\| \leq L_1 \|x_1 - x_2\|, \quad \textit{for some constant } L_1.$$

Then $\eta_{\theta^*} = \tilde{\eta}$.

Proof: First note that we are using histogram rule for classification. So, the the function $\eta_{\theta_k^*}$ is constant in each subsquare R_i. Let X_i be the center of each subsquare R_i. Then,

$$\eta_{\theta_k^*}(x) = \sum_{R_i, R_i \cap A_k \neq \phi} \eta_{\theta_k{}^*}(X_i) \mathbb{1}_{x \in R_i}$$

We will start with $\mathbb{E}[|\tilde{\eta}(X) - \eta_{\theta^*}(X)|]$ and we will bound this term in terms of r.

$$
\begin{aligned}
\mathbb{E}[|\tilde{\eta}(X) - \eta_{\theta^*}(X)|] &= \mathbb{E}[|\sum_{k=1}^{L} \eta_{\theta_k{}^*}(X)\mathbb{1}_{x\in A_k} - \eta_{\theta^*}(X)|] \\
&\leq \sum_{k=1}^{L} \mathbb{E}[|\eta_{\theta_k{}^*}(X) - \eta_{\theta^*}(X)|\mathbb{1}_{X\in A_k}] \\
&= \sum_{k=1}^{L} \mathbb{E}[|\eta_{\theta_k{}^*}(X) - \eta_{\theta^*}(X)|\,|\mathbb{1}_{X\in C_i}]\ \ \mathbb{P}(X \in A_k) \\
&\leq \sum_{k=1}^{L} \mathbb{E}[|\eta_{\theta_k{}^*}(X) - \eta_{\theta^*}(X)|\,|\mathbb{1}_{X\in C_i}] \\
&= \sum_{k=1}^{L} \mathbb{E}[|\sum_{R_i, R_i\cap A_k\neq\phi} \eta_{\theta_k{}^*}(X_i)\mathbb{1}_{x\in R_i} - \eta_{\theta^*}(X)|\,|\mathbb{1}_{X\in C_i}] \\
&\leq \sum_{k=1}^{L}\sum_{i=1}^{\frac{1}{r^2}} \mathbb{E}[|\eta_{\theta_k{}^*}(X_i) - \eta_{\theta^*}(X)|\mathbb{1}_{x\in R_i}\,|\mathbb{1}_{X\in C_i}] \\
&\leq \sum_{k=1}^{L}\sum_{i=1}^{\frac{1}{r^2}} \mathbb{E}[|\eta_{\theta_k{}^*}(X_i) - \eta_{\theta^*}(X)|\,\big|\mathbb{1}_{X\in A_k\mathbb{1}_{x\in R_i}}]\ \ \mathbb{P}(X \in R_i) \\
&= \sum_{k=1}^{L}\sum_{i=1}^{\frac{1}{r^2}} \mathbb{E}[|\eta_{\theta_k{}^*}(X_i) - \eta_{\theta^*}(X)|\,\big|\mathbb{1}_{X\in A_k\ \mathbb{1}_{x\in R_i}}]\ \ r^2 \\
&\leq \sum_{k=1}^{L}\sum_{i=1}^{\frac{1}{r^2}} L_1\,||X_i - X||\,r^2 \qquad \textit{Lipschitz Condition} \\
&\leq \sum_{k=1}^{L}\sum_{i=1}^{\frac{1}{r^2}} L_1\,\frac{\sqrt{2}r}{2}\,r^2 \\
&\leq \frac{1}{\sqrt{2}}\,L\,L_1\,r
\end{aligned}
$$

We can take $r = \frac{1}{n}$ where n is the number of labeled data points. As, we increase the number of labeled data points i.e. $n \to \infty$, or reduce the size of subsquares i.e. as $r \to 0$, the above expectation goes to 0 and we get the result. $\square$

6 Uncertainty Sampling

6.1 Introduction to Active Learning with Uncertainty Sampling

We implement Active Learning using another method known as *Uncertainty Sampling*. Uncertainty sampling is a core concept in active learning where the algorithm selectively queries the most uncertain instances for labeling, thereby optimizing the learning process with minimal labeled data. In this strategy, we identify and rank unlabeled training points where the model's predictions are least confident by calculating the value of $|\eta(x) - 0.5|$. These instances with the smallest margins are informative and then selected as candidates for querying.

6.2 Description Of the Algorithm

This algorithm is similar to the one described above but in this case, we use uncertainty sampling as described above to rank and then use them for sampling unlabeled data points

6.2.1 Uncertainty Sampling Function

The `uncertainty_sampling` function identifies the most uncertain instances for labeling based on the model's predictions. It operates as follows:

- **Input:**
  A trained model,dataset X, and the number of instances n_instances to query.
- **Process:**
  1. The function predicts probabilities of points being 1 for the dataset 'X'.
  2. It calculates the margins of uncertainty by finding the absolute difference between the probabilities and 0.5 i.e. $|\eta(x) - 0.5|$.
  3. It selects the indices of the `n_instances` instances with the smallest margins, indicating the highest uncertainty.
- **Output:** Indices of the queried instances.

6.2.2 Active Learning with Rejection Using Uncertainty Sampling

The `active_learning_with_rejection_US` function implements an active learning process with rejection based on uncertainty sampling.

Initialization

- We initialize lists `N`, `A`, ϵ, η, λ, and D.
- We set the initial number of samples N_1 equal to `B`.
- We select an initial random subset of training data `'X0, Y0'` and train the initial model η_0.

Parameters and Variables:

- We calculate the initial values for N_k (number of samples) and ϵ_k (error rate).

Active Learning Loop:

We loop until the budget $B + N_k * \epsilon_k$ is less than the total available training data `'N1'`:

1. **Uncertainty Sampling:** We identify the most uncertain instances from the dataset `A` using the `uncertainty_sampling` function.
2. **Threshold Calculation:** We compute the rejection threshold λ_k for the selected uncertain instances.
3. **New Uncertain Region:**
   We then find the new uncertain region based on the calculated threshold and store it in list `A`.
4. **Random Sampling and Training:** We randomly select a subset of points of size $N_k\epsilon_k$ from new region points for the current iteration along with their labels and train a new model η_k on those points.
5. **Update Variable :**

- We update the value of N_k and ϵ_k as we did in the previous algorithm. We also update the list of classifiers η at each iteration.
- We also calculate the time taken at each step of the algorithm

Output :

We return the list of error rates ϵ,list of uncertain regions `A`, list of classifiers η and list of time taken for each step of all iterations.

6.2.3 Labeling Points Function

The `label_points` function labels new instances using the trained models and datasets from the active learning process. The steps are as follows:

Input: Test dataset X_test, uncertain regions `A`, and models η.

Process:

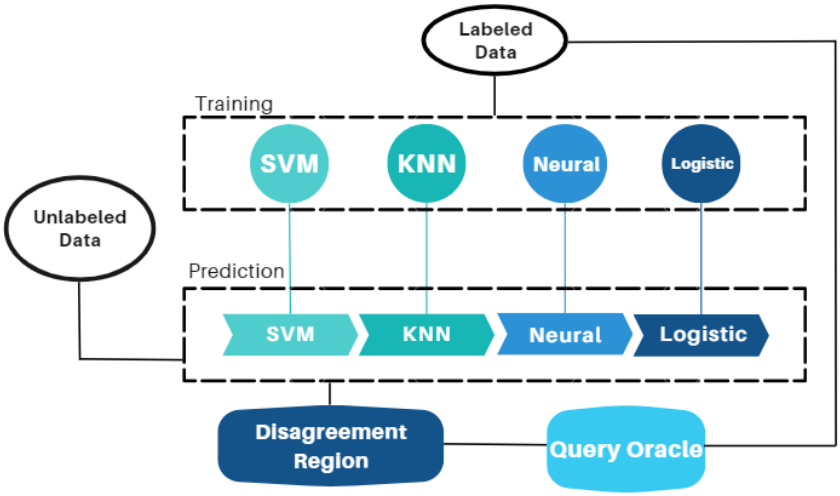
- We convert each sublist in `A` to a NumPy array.
- We initialize arrays for storing prediction values and labels.
- We iterate through the models and datasets, updating the prediction values for instances that belong to the current dataset and not the next. For the last dataset, we update prediction values for instances that belong to it.
- We label instances based on whether their predicted probability values of being 1 are above or below 0.5.

Output: Prediction values and labels for the test dataset.

7 Query By Committee

7.1 Introduction to Active Learning with Query By Committee

Now, we implement another strategy for Active Learning to optimize the sampling of unlabeled data points known as *Query By Committee*. The whole process is almost similar except for the sampling of unlabeled data points. We create a set of five classifiers and employ entropy-based sampling to select instances for labeling. We train all classifiers on a set of labeled points. At each iteration k, we select `M` points from the previous uncertain region where disagreement occurs between members of the committee. These points are the "informative points" that then help determine the next uncertain region based on these points.



7.2 Description Of the Algorithm

This algorithm is similar to the ones described above but here we use *Disagreement Sampling* to sample the unlabeled data points to calculate the threshold for the new uncertain region.

7.2.1 Vote Entropy Function

The `vote_entropy` function calculates the entropy of votes from a committee of classifiers for each instance in a given dataset. When predictions are made by the committee of classifiers for an instance, it has a corresponding probability distribution of labels to it. We then calculate Entropy $-p\sum\log(p)$ which is a measure of uncertainty or disagreement among the committee members, which is crucial for identifying informative instances. Higher entropy indicates higher disagreement between classifiers for that instance.

Input

- **committee:** A list of trained classifiers.
- `X`: The dataset for which the entropy is to be calculated.

Process

- Each classifier in the committee predicts the labels for the instances in `X`.
- The predictions are counted to obtain a frequency of labels for each class.
- Probabilities are computed by normalizing the frequencies.
- Entropy is calculated using the formula $-p\sum log(p)$

Output

- An array of entropy values for each instance in `X`

7.2.2 `query_by_committee_sampling` Function

This function uses the entropy values to select a subset of instances that the committee is most uncertain about. This process is also known as disagreement sampling, where instances with the highest disagreement among the committee members are selected for labeling.

Input:

- **committee**: A list of trained classifiers.
- **X:** The dataset to sample from.
- **n_instances:** The number of instances to query.

Process:

- Entropy values for X are calculated using the `vote_entropy` function.
- Instances are sorted by their entropy values, and the top `n_instances` with the highest entropy are selected.

Output:

- Indices of the selected instances.

7.2.3 `fit_classifier` Function

A utility function to train a classifier on a given dataset.

Input:

- **clf**: The classifier to be trained.
- **X,y**: The training data and corresponding labels.

Output:

- The trained classifier

7.2.4 `active_learning_with_rejection_QBC` Function

This is the main function implementing the active learning loop with query-by-committee (QBC) sampling and disagreement sampling. This is same as the previous two described above.

Input:

- **data:** Initial data containing the pool of instances and labels which is representative of actual instance space.
- **X_train, Y_train:** Training data and corresponding labels.
- β**, N1, d, c, M:** Parameters controlling the active learning process (same as defined in previous algorithm).

Process:

- Initialization of various parameters including the committee, labeled and unlabeled sets, and thresholds.
- η_0 is trained on a subset of X_train and its corresponding labels.
- In each iteration :
  - The entropy-based sampling (disagreement sampling) selects `M` unlabeled instances for querying.
  - We determine threshold using a predefined function (`find_threshold`).
  - We then find a new uncertain region based on that threshold.
  - We use data points from the new uncertain region and ask for their labels for sampling.
  - A new committee of classifiers is trained on the updated training set.
  - The process continues until the budget constraint (N1) is met.

Output:

- Lists containing the error rates, uncertain regions, trained committees, and time taken for each step of each iteration.

7.2.5 `label_points` Function

This function works exactly in the same manner as defined in the above algorithms. Since we have a set of classifiers here for each region, we take a mean of probabilities for an instance and then assigns label.

Input:

- **X_test:** Test dataset.
- **A:** List of instance subsets from each iteration.
- η**:** List of committees from each iteration

Process:

- For each instance, we check in which uncertain region it belongs and then use the classifiers trained on that region to predict the probabilities.
- We then assign labels based on the aggregated predictions similarly as defined in the previous methods.

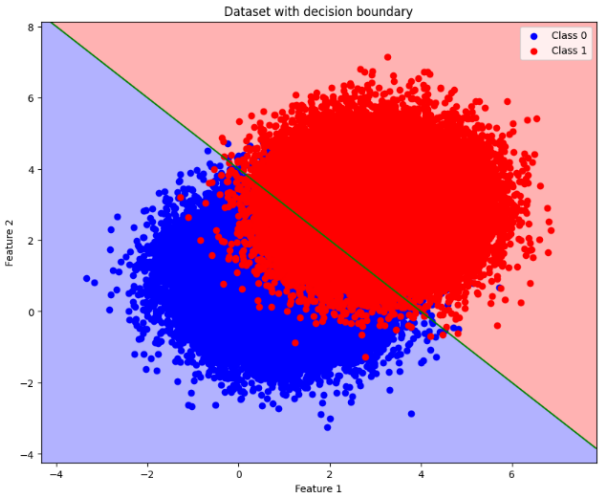
Output:

- **eta_values:** Aggregated prediction probabilities.
- **labels:** Final predicted labels.

8 Numerical Experiments

8.1 Synthetic Data

Now, we aim to implement Active Learning Algorithm. For this purpose, we simulate a binary classification problem, we generated synthetic data using the approach identical to that described in Section 4.1.



Setting:

Our goal is to compare the performance of different classifiers in the active and passive learning settings. To get an idea of the performance, we perform 20 experiments. At each iteration, we split data into train and test sets, implement the active and passive algorithms, and then calculate the accuracy for each algorithm. At the end, we calculate the mean accuracy for each algorithm.

Here, we calculate the results for Active Learning with the rejection method and uncertainty sampling using Logistic Regression, K-Nearest Neighbors (KNN) and Decision Tree classifier. For QBC, we use a committee of five classifiers- Random Forest, Gradient Boosting, Support Vector Classifier, Logistic Regression, and Decision Tree.

We generated 10^5 points and used it as our instance space $\mathcal{X} = [0, 1]^2$. We have a budget of $N = 400$ points. Other parameters of the algorithm are set according to theorem 5. We use 3000 test points to calculate the accuracy.

Results:

The results are as follows -

Accuracy for QBC : 0.9158 ± 0.0046

Accuracy for other classifiers:

Classifier	N	Active Framework	Passive Framework	Uncertainty Sampling
Logistic Regression	400	0.9186 ± 0.0026	0.9187 ± 0.0027	0.9165 ± 0.0019
SVC	400	0.9165 ± 0.0043	0.9185 ± 0.0023	0.9169 ± 0.0034
Decision Tree	400	0.8734 ± 0.0222	0.8740 ± 0.0111	0.8781 ± 0.0202

Table 1: Comparison of different classifiers and frameworks.

Learning Curves:

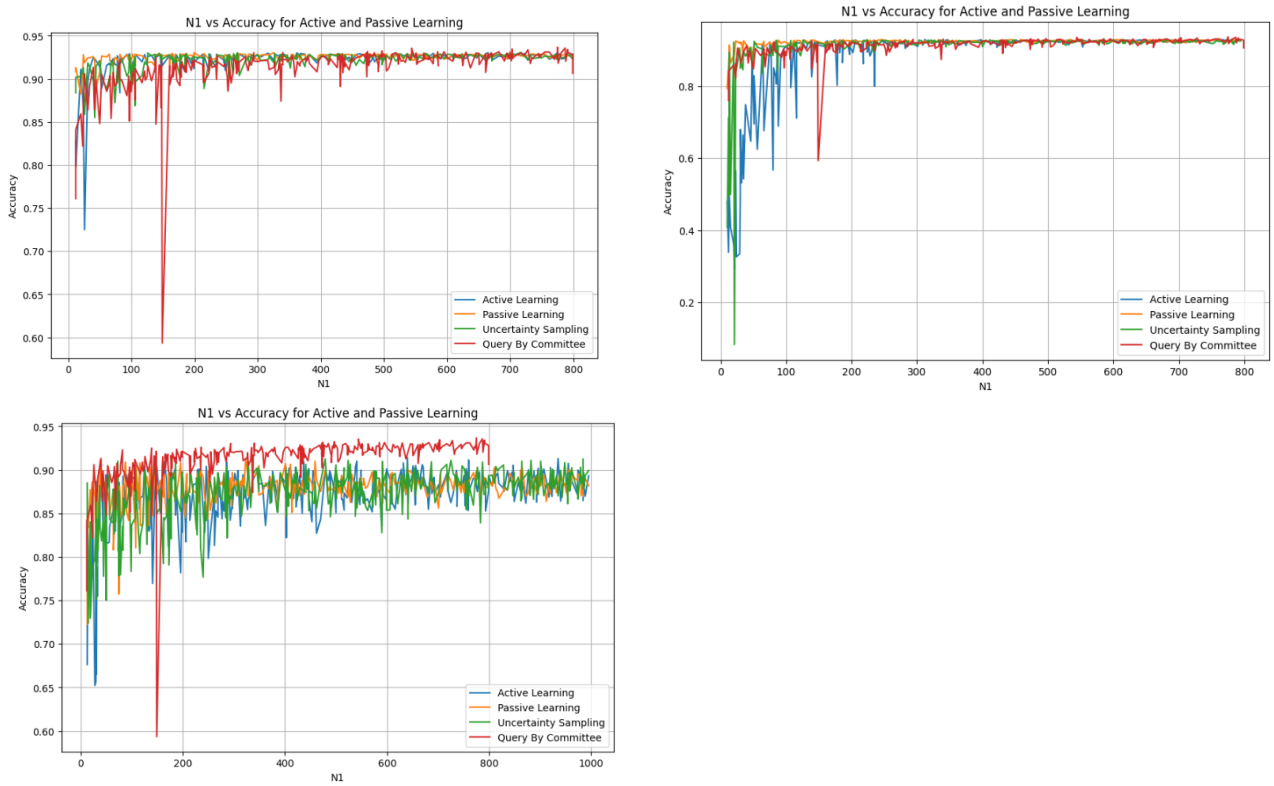


Table 2: (a)Logistic Regression (b) SVM (c)Decision Tree

Observations:

- All methods generally show an improvement in accuracy as the budget ($N1$) increases.
- The accuracy tends to stabilize after a certain number of labeled instances, indicating a saturation point beyond which additional data does not significantly improve performance.
- All methods converge to a similar accuracy level (around 0.90-0.95).
- Query By Committee shows more pronounced fluctuations early on compared to the other methods. This might be due to involves multiple models (the "committee") making decisions on which samples to label. The diversity among these models can lead to inconsistent choices early on.
- Across all models, Active Learning, Passive Learning, Uncertainty Sampling, and Query By Committee converge to similar accuracy levels, indicating the robustness of these methods.

8.2 Non-synthetic Dataset (Stroke Prediction)

This dataset is used to predict whether a patient is likely to get a stroke based on the input parameters like gender, age, various diseases, and smoking status. We first prepare the dataset for training and drop some irrelevant columns. We convert categorical columns into numerical data using `LabelEncoder`. Then, we perform data normalization to have our instance space in $[0, 1]^d$. We take all the points of the dataset as our whole instance space. We use the same setting as used in the case of the synthetic dataset, perform the same experiment again, and employ the same algorithms to calculate the mean accuracy.

Here, we have a total of 5110 data points and 9 features in the dataset. We use a budget of 500 points and around 1500 test points to calculate the accuracy.

Results:

The results for the non-synthetic dataset are as follows-

Accuracy for QBC : 0.9511 ± 2.220e-16

Classifier	N	Active Framework	Passive Framework	Uncertainty Sampling
Logistic Regression	500	0.9508 ± 0.0012	0.9511 ± 0.0004	0.9511 ± 2.220e-16
SVC	500	0.9509 ± 0.0008	0.9511 ± 2.220e-16	0.9511 ± 2.220e-16
Decision Tree	500	**0.9152 ± 0.0141**	0.9056 ± 0.0097	0.9511 ± 2.220e-16

Table 3: Comparison of different classifiers and frameworks.

Learning Curves:

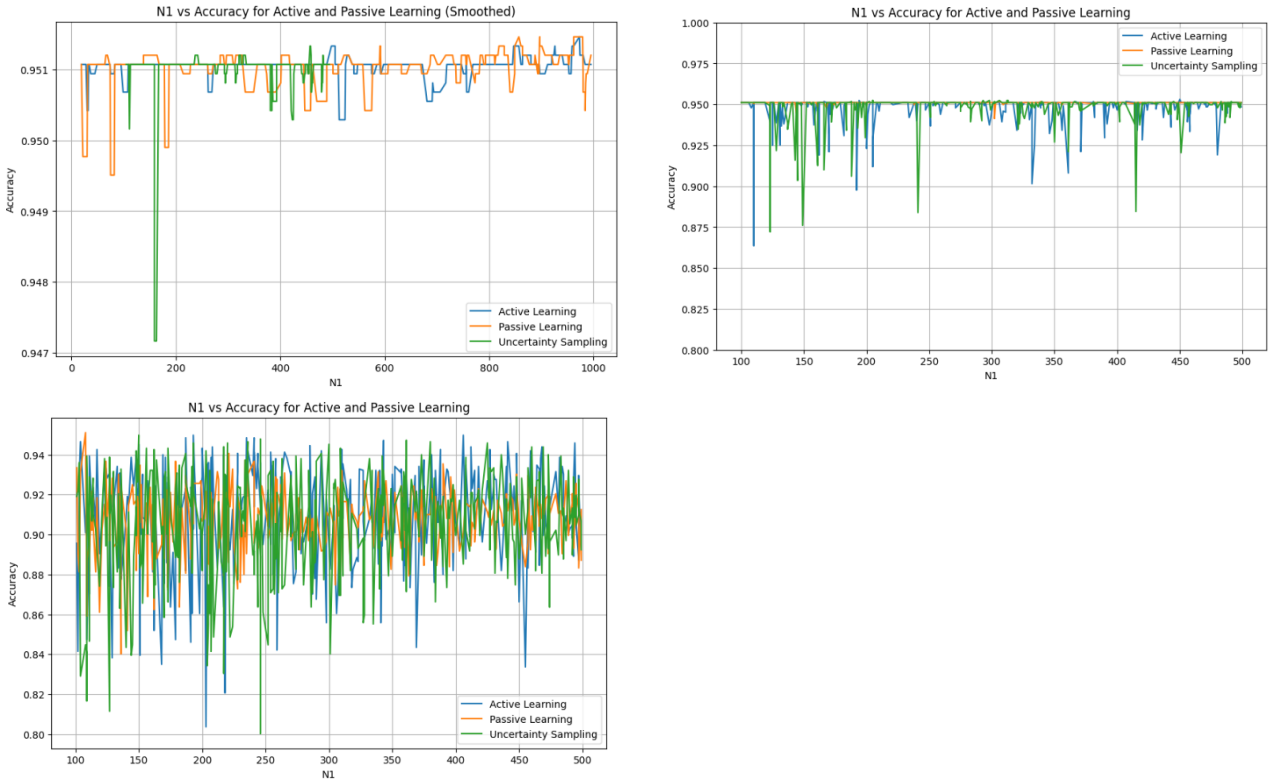


Table 4: (a)Logistic Regression (b) SVM (c)Decision Tree

Observations:

- The accuracy for all methods (Active Learning, Passive Learning, and Uncertainty Sampling) generally shows stability and less fluctuation compared to the synthetic dataset.
- All methods converge to similar accuracy levels across different models, indicating the robustness of the learning strategies on non-synthetic data.
- We can also observe from the table that Active classifier performs slightly better than the passive one in the case of Decision Tree.
- It can also be observed from the table that Uncertainty Sampling and QBC has less standard deviation compared to other methods.

9 Comparison of Active Learning Approaches Using modAL and Alipy

In this section, we evaluate and compare the performance of two popular Python libraries for active learning: **modAL** and **Alipy**. Both libraries provide flexible and efficient tools for implementing active learning strategies, allowing for the iterative selection of the most informative data points to label and train models.

9.1 Experimental Setup

We use the same synthetic and non-synthetic dataset (Stroke Prediction Dataset) to compare the results.

The dataset was initially split into training and test sets, with a small portion of the training data (20 samples) randomly selected as the initial labeled set. The active learning process was carried out using uncertainty sampling, where the model queries the data points it is most uncertain about.

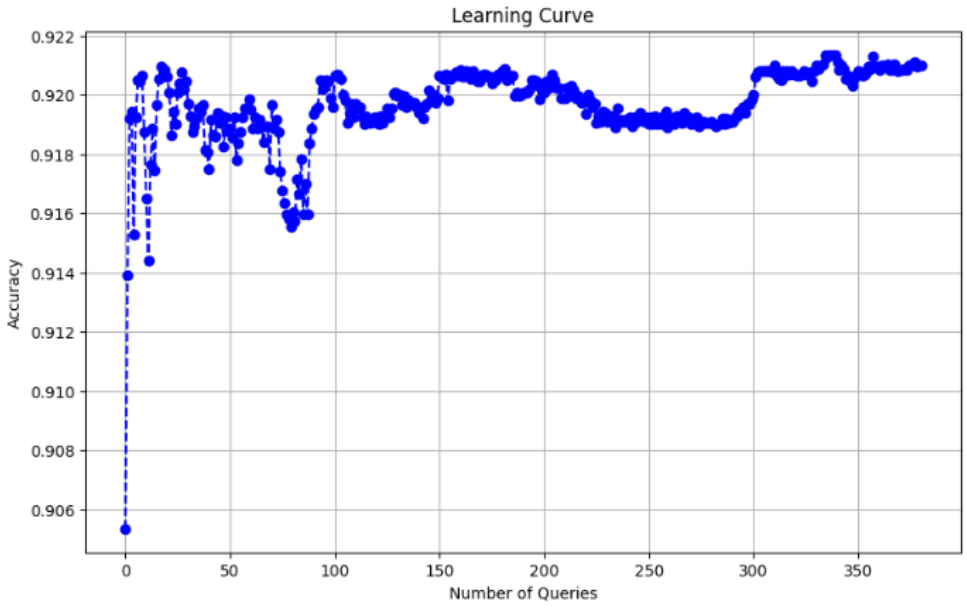
For modAL, a logistic regression model was used as the base classifier. The active learning loop was executed for 380 queries, with the model retrained on each newly labeled instance. The performance was evaluated using the accuracy metric on the test set after each query, and a learning curve was plotted to visualize the accuracy improvement over time.

For `Alipy`, we initialized the active learning experiment with similar settings but limited the number of queries to 50 due to the time-intensive nature of the process. The QueryInstanceUncertainty strategy with the least_confident measure was employed. The performance was tracked throughout the active learning process, with the final accuracy recorded after 50 queries.

9.2 Results for Synthetic Dataset

`ModAl`

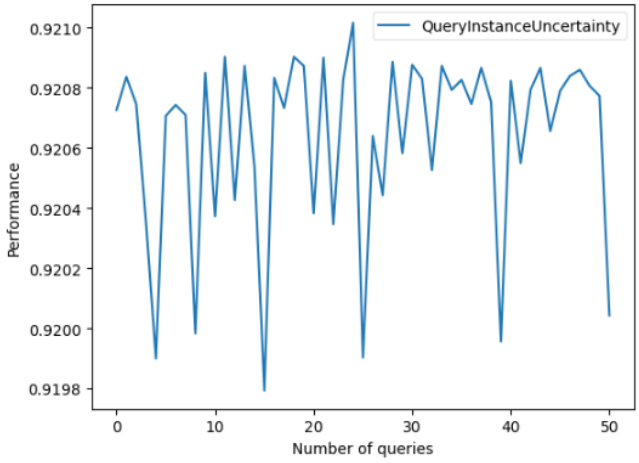
The final accuracy obtained after implementation is **0.921** and the learning curve is as follows -



`Alipy`

The results for `Alipy` is as follows -

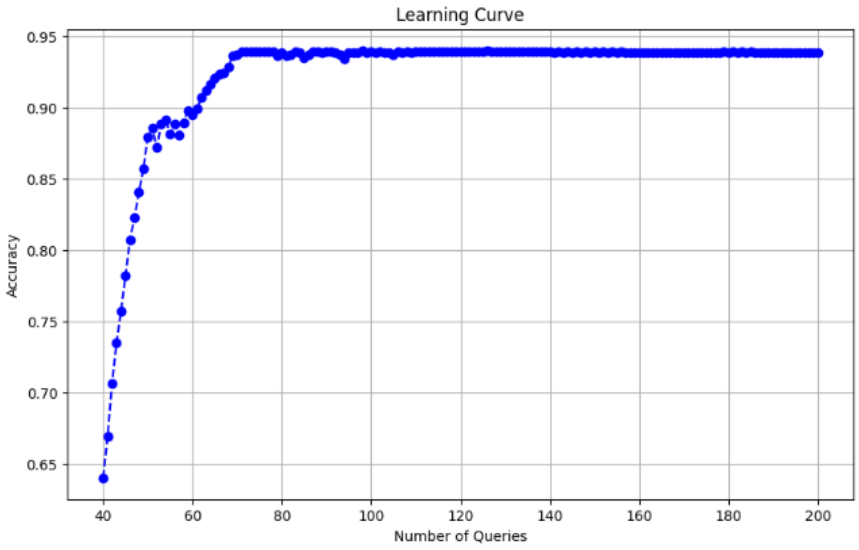
Methods	number_of_queries	number_of_different_split	performance	batch_size
QueryInstanceUncertainty	50	10	0.921 ± 0.00	1



9.3 Results for Non-Synthetic Dataset

`ModAl`

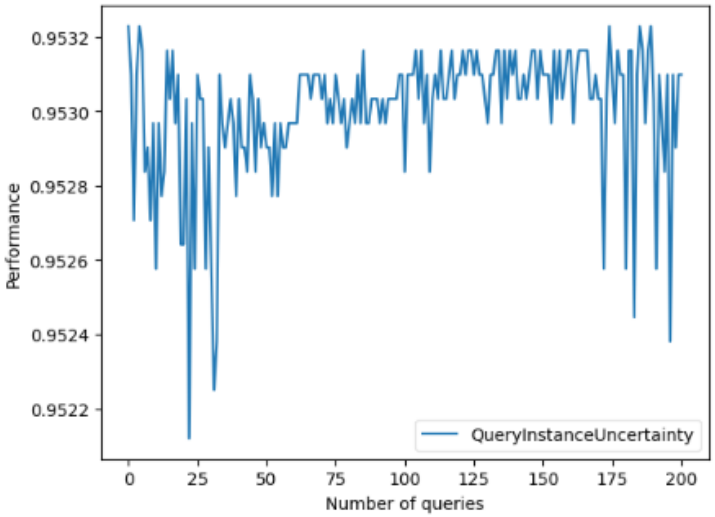
The final accuracy obtained after implementation is **0.938** and the learning curve is as follows -



`Alipy`

The results for `Alipy` is as follows -

Methods	number_of_queries	number_of_different_split	performance	batch_size
QueryInstanceUncertainty	200	10	0.953 ± 0.00	1



10 Active Learning for Imbalanced Datasets

Here, I refer to the paper titled "Minority Class Oriented Active Learning for Imbalanced Datasets". addresses the challenge of active learning (AL) in the context of imbalanced datasets, which are common in real-world applications. Traditional AL methods tend to assume that the data is balanced, which can lead to poor performance when applied to imbalanced datasets. The authors propose a novel AL strategy designed to address this issue by focusing on improving the representation of minority classes.

Problem Formalization

Imbalanced datasets are characterized by a significant disparity between the number of instances in different classes, typically with one or more classes (minority classes) being underrepresented. This imbalance poses challenges for machine learning models, which may become biased towards the majority class. The paper highlights the limitations of existing AL methods in handling such imbalances and the need for an approach that can effectively target minority classes.

The process begins by defining an unlabeled dataset D^U consisting of samples $x_i \in \mathcal{X}, i = \{1, \dots u\}$ which are i.i.d realiztions. The AL process is iterative, where a small budget $b < u$ is allocated for labeling data across t iterations. Initially, a small labeled set of random realizations is drawn to form ${D^L}_0$ to train model M_0

At each iteration, a batch of samples of size $\frac{b}{t}$ is selected using an acquisition function (that uses the information provided by M_{k-1}) from ${D^U}_k = D^U \backslash {D^L}_{k-1}$ to update labeled subset ${D^L}_k$ which is then used to train M_k

Two training methods are explored:

1. **Deep Model Update:** Inspired by typical deep learning AL processes, this approach involves fine-tuning the model at each iteration using M_S and D_L^k.

2. **Transfer Learning with Shallow Classifiers:** In this approach, a pre-trained model M_S provides feature embeddings that are used to train shallow classifiers during AL iterations, assuming knowledge from M_S can be transferred to D^U.

A switch criterion based on cross-validation is proposed to decide when to switch from deep model updating to transfer learning during AL, ensuring the best performance across different stages of the AL process.

The following four types of Aquistion processes are discussed here -

1. **Random Sampling** This baseline involves randomly selecting samples for annotation, which serves as a simple but commonly used method in AL. It is often used as a reference to evaluate the effectiveness of more sophisticated sampling strategies.
2. **Margin Sampling** This method selects samples based on the uncertainty of the model's predictions. Specifically, it focuses on samples where the model's top two predicted class probabilities are closest, indicating high uncertainty. The idea is that labeling such samples can potentially improve the model's performance in uncertain regions.
3. **Corrected Sampling** Recently introduced, this method selects samples by considering both the informativeness and representativeness of the data points. It aims to optimize the coverage of the data's feature space. Samples are chosen based on how much they can reduce the distance between the labeled and unlabeled data points in feature space.
4. **Uncertainty Diversity Sampling** This method combines the uncertainty of predictions with a diversity criterion. It selects the most uncertain samples while also ensuring that these samples are diverse. The goal is to reduce redundancy and enhance the effectiveness of the labeled data.

Each of these baselines represents a different strategy for sample selection in AL, ranging from simple random selection to more complex methods that aim to maximize learning efficiency by focusing on uncertainty, diversity, or data representativeness.

11 Conclusion

11.1 Contributions:

1. **Theoretical :**
   - Read the research paper *Active learning algorithm through the lens of rejection arguments* by Christophe Denis, Mohamed Hebiri, Boris Njike, Xavier Siebert (2022).
   - Proof of the theorem for Empirical Risk Minimization for passive learning framework.
2. **Practical :**
   - Simulated synthetic datasets and studied the results on passive learning framework and compared them with Bayes classifier
   - Implementation Of active learning algorithm using rejection method, Query by Committee (QBC), and Uncertainty Sampling from scratch.
   - Implementation of the above algorithms on Synthetic and Real Dataset (Stroke Prediction).
   - Comparison of the active learning results with passive learning algorithms.

11.2 Future Perspective

1. **Theoretical:**
   - Understand the theoretical part of the paper concerning the bound on the emperical risk.
   - Propose a similar result but in the case of parametric active learning setting
2. **Practical:**
   - Find new scenario where active learning framework is relevant.
   - Comparison of the results of active learning algorithms implemented with already existing active learning libraries online.

12 Appendix

12.1 Psuedocode for Active Learning with Rejection Method

Algorithm 1 Active Learning with Rejection

function SCORE_FUNCTION(X, η)
    probs ← PREDICT_PROBA(η, X)[:, 1]
    scores ← MAXIMUM(probs, 1 - probs)
    return scores
end function
function FIND_THRESHOLD(A_k, η, p)
    probs ← PREDICT_PROBA(η, A_k)[:, 1]
    scores ← MAXIMUM(probs, 1 - probs)
    threshold ← QUANTILE(scores, p)
    return threshold
end function

Algorithm 2 Label Points

function LABEL_POINTS(X_{test}, A, η)
    A_arrays ← [ARRAY(sublist) for sublist in A]
    η_values ← ZEROS(len(X_{test}))
    labels ← ZEROS(len(X_{test}), dtype=int)
    for j in RANGE(len(η) - 1) **do**
        $mask\_j$ ← ALL({}ISIN(X_{test}, $A\_arrays[j]$), axis=1} & $\sim$ ALL({}ISIN(X_{test}, $A\_arrays[j+1]$), axis=1}
        $\eta\_values[mask\_j]$ += PREDICT_PROBA({}$\eta[j]$, $X_{\text{test}}[mask\_j]$}[:, 1]
    end for
    $mask\_last$ ← ALL({}ISIN()X_{test}, A_arrays[len(A) − 1], axis=1}
    $\eta\_values[mask\_last]$ += PREDICT_PROBA(η[len(A) − 1], $X_{\text{test}}[mask\_last]$)[:, 1]
    labels ← $(\eta\_values \geq 0.5).astype(int)$
    return $\eta\_values, labels$
end function

Algorithm 3 Active Learning with Rejection

function ACTIVE_LEARNING_WITH_REJECTION($data$, X_{train}, Y_{train}, $\beta = 1$, $N_1 = 500$, $d = 2$, $c = 1.5$, $M = 100$)
  $N \leftarrow [\lfloor \sqrt{N_1} \rfloor]$
  $A \leftarrow [data[0]]$
  $\epsilon \leftarrow [1]$
  $\eta, \lambda, D, time\_taken \leftarrow [], [], [[]], []$
  $B \leftarrow N[0]$
  $L \leftarrow [data[1]]$
  $k, i \leftarrow 1, 1$
  indices $\leftarrow$ RANDOM_CHOICE(len(X_{train}), size=$N[0]$, replace=False)
  $X_0, Y_0 \leftarrow X_{\text{train}}[indices], Y_{\text{train}}[indices]$
  $\eta_0 \leftarrow$ LOGISTICREGRESSION
  FIT(η_0, X_0, Y_0)
  $\eta \leftarrow [\eta_0]$
  $D[0] \leftarrow$ ZIP(X_0, Y_0)
  $\delta \leftarrow 0.46$
  $N.append(\lfloor c \cdot N[i-1] \rfloor)$
  $\epsilon.append(\min(0.85, (\log(N_1/\delta)) \cdot (\log(N_1)) \cdot (N[i-1]^{-\beta/(2\beta+d)})))$
  $N_k \leftarrow N[1]$
  $e_k \leftarrow \epsilon[1]$
  $\lambda.append(1)$
  while $B + \lfloor N_k \cdot e_k \rfloor \leq N_1$ **do**
    $t1 \leftarrow$ TIME
    indices1 $\leftarrow$ RANDOM_CHOICE(len($A[i-1]$), size=M, replace=False)
    $D_U \leftarrow [A[i-1][j]$ for j in $indices1]$
    $\lambda.append($FIND_THRESHOLD(D_U, $\eta[i-1]$, e_k))
    $t2 \leftarrow$ TIME
    scores $\leftarrow$ SCORE_FUNCTION() ARRAY()$A[i-1]$,$\eta[i-1]$
    mask $\leftarrow scores \leq \lambda[i]$
    $A.append($ARRAY()$A[i-1]$[mask].tolist())
    $L.append($ARRAY()$L[i-1]$[mask].tolist())
    $t3 \leftarrow$ TIME
    indices $\leftarrow$ RANDOM_CHOICE(l)en($L[i]$), size=$\lfloor N_k \cdot e_k \rfloor$, replace=False
    d_points $\leftarrow [(x, y)$ for x, y in ZIP($\{$)$A[i], L[i]\}]$
    $D.append([d\_points[h]$ for h in $indices])$
    $X_i \leftarrow [pair[0]$ for $pair$ in $D[i]]$
    $Y_i \leftarrow [pair[1]$ for $pair$ in $D[i]]$
    $t4 \leftarrow$ TIME
    $\eta_i \leftarrow$ LOGISTICREGRESSION
    FIT()η_i, X_i, Y_i
    $\eta.append(\eta_i)$
    $t5 \leftarrow$ TIME
    $B \leftarrow B + \lfloor N_k \cdot e_k \rfloor$
    $i \leftarrow i + 1$
    $N.append(\lfloor c \cdot N[i-1] \rfloor)$
    $\epsilon.append(\min(0.85, (\log(N_1/\delta)) \cdot (\log(N_1)) \cdot (N[i-1]^{-\beta/(2\beta+d)})))$
    $N_k \leftarrow N[i]$
    $e_k \leftarrow \epsilon[i]$
    $t6 \leftarrow$ TIME
    $time\_taken.append([t2 - t1, t3 - t2, t4 - t3, t5 - t4, t6 - t5, t6 - t1])$
  end while
  return ϵ, A, η, $time\_taken$
end function

12.2 Psuedocode for Active Learning with Uncertainty Sampling

Algorithm 4 Uncertainty Sampling

```
function UncertaintySampling(model, X, n_instances=10)
    probs ← model.predict_proba(X)[:, 1]
    margins ← abs(probs - 0.5)
    query_indices ← argsort(margins)[:n_instances]
    return query_indices
end function
```

Algorithm 5 Label Points

```
function LabelPoints(X_test, A, eta)
    A_arrays ← [array(sublist) for sublist in A]
    eta_values ← zeros(len(X_test))
    labels ← zeros(len(X_test), dtype=int)
    for j in range(len(eta) - 1) do
        mask_j ← all(isin(X_test, A_arrays[j]), axis=1) and not all(isin(X_test, A_arrays[j + 1]), axis=1)
        eta_values[mask_j] += eta[j].predict_proba(X_test[mask_j])[:, 1]
    end for
    mask_last ← all(isin(X_test, A_arrays[-1]), axis=1)
    eta_values[mask_last] += eta[-1].predict_proba(X_test[mask_last])[:, 1]
    labels ← (eta_values >= 0.5).astype(int)
    return eta_values, labels
end function
```

Algorithm 6 Active Learning with Rejection using Uncertainty Sampling

```
function ACTIVELEARNINGWITHREJECTIONUS(data, X_train, Y_train, beta=1, N1=200, d=2, c=1.1, M=100)
    N ← [sqrt(N1)]
    A ← [data[0]]
    epsilon ← [1]
    eta ← []
    lambda ← []
    D ← [[]]
    B ← N[0]
    L ← [data[1]]
    k ← 1
    time_taken ← []
    indices ← random_choice(len(X_train), size=N[0], replace=False)
    X0, Y0 ← X_train[indices], Y_train[indices]
    eta_0 ← LogisticRegression()
    eta_0.fit(X0, Y0)
    eta.append(eta_0)
    D[0] ← list(zip(X0, Y0))
    delta ← 0.46
    i ← 1
    N.append(int(c * N[i - 1]))
    epsilon.append(min(0.85, (log(N1 / delta)) * (log(N1)) * ((N[i - 1]) ** (-beta / (2 * beta + d)))))
    N_k ← N[1]
    e_k ← epsilon[1]
    lambda.append(1)
    while B + floor(N_k * e_k) ≤ N1 do
        t1 ← time()
        D_U_indices ← UncertaintySampling(eta[i - 1], A[i - 1], M)
        D_U ← [A[i - 1][j] for j in D_U_indices]
        lambda.append(find_threshold(D_U, eta[i - 1], e_k))
        t2 ← time()
        scores ← score_function(array(A[i - 1]), eta[i - 1])
        mask ← scores ≤ lambda[i]
        A.append(array(A[i - 1])[mask].tolist())
        L.append(array(L[i - 1])[mask].tolist())
        t3 ← time()
        indices ← random_choice(len(L[i]), size=min(len(L[i]), int(floor(N_k * e_k))), replace=False)
        d_points ← [(x, y) for x, y in zip(A[i], L[i])]
        D.append([d_points[h] for h in indices])
        X_i ← [pair[0] for pair in D[i]]
        Y_i ← [pair[1] for pair in D[i]]
        t4 ← time()
        eta_i ← LogisticRegression()
        eta_i.fit(X_i, Y_i)
        eta.append(eta_i)
        t5 ← time()
        B += floor(N_k * e_k)
        i += 1
        N.append(int(c * N[i - 1]))
        epsilon.append(min(0.85, (log(N1 / delta)) * (log(N1)) * ((N[i - 1]) ** (-beta / (2 * beta + d)))))
        N_k ← N[i]
        e_k ← epsilon[i]
        t6 ← time()
        time_taken.append([t2-t1, t3-t2, t4-t3, t5-t4, t6-t5, t6-t1])
    end while
    return epsilon, A, eta, time_taken
end function
```

12.3 Pseudocode for Active Learning with Query By Committee (QBC)

Algorithm 7 vote_entropy

Require: $committee$, X
Ensure: $vote\_entropy$
$committee\_votes \leftarrow$ np.array($[clf.predict(X)$ for clf in committee$]$)
$bincounts \leftarrow$ np.apply_along_axis(np.bincount, 0, committee_votes, minlength=2)
$probabilities \leftarrow bincounts/$len(committee)
$vote_entropy \leftarrow -$np.sum$(probabilities *$ np.log$(probabilities + 1e - 10)$, axis $= 0)$
return $vote\_entropy$

Algorithm 8 query_by_committee_sampling

Require: $committee$, X, $n\_instances = 10$
Ensure: $query\_indices$
$entropy \leftarrow$ vote_entropy$(committee, X)$
$query\_indices \leftarrow$ np.argsort$(entropy)[-$n_instances :$]$
return $query\_indices$

Algorithm 9 fit_classifier

Require: clf, X, y
Ensure: $trained\_classifier$
return $clf.fit(X, y)$

Algorithm 10 label_points

Require: X_test, A, η
Ensure: η_values, $labels$
$A\_arrays \leftarrow$ [np.array(sublist) for sublist in A]
$\eta\_values \leftarrow$ np.zeros(len(X_test))
$labels \leftarrow$ np.zeros(len$(X_test), dtype = int)$
for j in range(len$(\eta) - 1$) **do**
    $mask\_j \leftarrow$ np.all(np.isin$(X_test, A_arrays[j]),$ axis $= 1)\& \sim$ np.all(np.isin$(X_test, A_arrays[j + 1]),$ axis $= 1)$
    $\eta\_values[mask\_j] \leftarrow \eta\_values[mask\_j] +$ np.mean$([clf.predict_proba(X_test[mask_j])[:, 1]$ for clf in $\eta[j]],$ axis $= 0)$
end for
$mask\_last \leftarrow$ np.all(np.isin$(X_test, A_arrays[-1]),$ axis $= 1)$
$\eta\_values[mask\_last] \leftarrow \eta\_values[mask\_last] +$ np.mean$([clf.predict_proba(X_test[mask_last])[:, 1]$ for clf in $\eta[-1]],$ axis $= 0)$
$labels \leftarrow (\eta\_values \geq 0.5).astype(int)$
return η_values, $labels$

Algorithm 11 active_learning_with_rejection_QBC

Require: $data$, X_train, Y_train, $\beta = 1$, $N1 = 200$, $d = 2$, $c = 1.1$, $M = 100$
Ensure: ϵ, A, η, $time\_taken$

$N \leftarrow$ [int(np.sqrt($N1$))]
$A \leftarrow [data[0]]$
$\epsilon \leftarrow [1]$
$\eta \leftarrow []$
$\lambda \leftarrow []$
$D \leftarrow [[]]$
$B \leftarrow N[0]$
$L \leftarrow [data[1]]$
$k \leftarrow 1$
$time\_taken \leftarrow []$
$indices \leftarrow$ np.random.choice(len(X_train), size $= N[0]$, replace=False)
$X0, Y0 \leftarrow X\_train[indices], Y\_train[indices]$
$committee \leftarrow [$
    RandomForestClassifier(random_state=42),
    GradientBoostingClassifier(random_state=42),
    SVC(probability=True, random_state=42),
    LogisticRegression(random_state=42),
    DecisionTreeClassifier(random_state=42)
]
for each clf in $committee$ **do**
    $clf.fit(X0, Y0)$
end for
$\eta.append(committee)$
$D[0] \leftarrow$ list(zip($X0, Y0$))
$\delta \leftarrow 0.46$
$i \leftarrow 1$
$N.append($int$(c * N[i-1]))$
$\epsilon.append($min$(0.85, ($np.log$(N1/\delta)) * ($np.log$(N1)) * ((N[i-1]) * *(-\beta/(2 * \beta + d)))))$
$N_k \leftarrow N[1]$
$e_k \leftarrow \epsilon[1]$
$\lambda.append(1)$

Algorithm 12 active_learning_with_rejection_QBC (continued)

while B + np.floor($N_k * e_k$) $\leq N1$ **do**
  $t1 \leftarrow$ time.time()
  $D_U\_indices \leftarrow$ query_by_committee_sampling($\eta[i-1], A[i-1], M$)
  $D_U \leftarrow [A[i-1][j]$ for j in $D_U\_indices]$
  $\lambda.append$(find_threshold($D_U, \eta[i-1][0], e_k$))
  $t2 \leftarrow$ time.time()
  $scores \leftarrow$ score_function(np.array($A[i-1]$), $\eta[i-1][0]$)
  $mask \leftarrow scores \leq \lambda[i]$
  $A.append$(np.array($A[i-1]$)$[mask].tolist()$)
  $L.append$(np.array($L[i-1]$)$[mask].tolist()$)
  $t3 \leftarrow$ time.time()
  $indices \leftarrow$ np.random.choice(len($L[i]$), size=min(len($L[i]$), int(np.floor($N_k * e_k$))), replace=False)
  $d_points \leftarrow [(x, y)$ for x, y in zip($A[i], L[i]$)]
  $D.append([d_points[h]$ for h in indices])
  $X_i \leftarrow$ [pair[0] for pair in $D[i]$]
  $Y_i \leftarrow$ [pair[1] for pair in $D[i]$]
  $t4 \leftarrow$ time.time()
  $new\_committee \leftarrow$ [
  RandomForestClassifier(random_state=42),
  GradientBoostingClassifier(random_state=42),
  SVC(probability=True, random_state=42),
  LogisticRegression(random_state=42),
  DecisionTreeClassifier(random_state=42)
]
  for each clf in $new\_committee$ **do**
    $clf.fit(X_i, Y_i)$
  end for
  $\eta.append(new\_committee)$
  $t5 \leftarrow$ time.time()
  $B \leftarrow B +$ np.floor($N_k * e_k$)
  $i \leftarrow i + 1$
  $N.append$(int($c * N[i-1]$))
  $\epsilon.append$(min(0.85, (np.log($N1/\delta$)) * (np.log($N1$)) * (($N[i-1]$) * *($-\beta/(2 * \beta + d)$)))))
  $N_k \leftarrow N[i]$
  $e_k \leftarrow \epsilon[i]$
  $t6 \leftarrow$ time.time()
  $time\_taken.append([t2-t1, t3-t2, t4-t3, t5-t4, t6-t5, t6-t1])$
end while
return ϵ, A, η, $time\_taken$

13 Bibliography

1. Linear vs. Non-Linear Classification
2. Active Learning Sampling Strategies
3. Christophe Denis, Mohamed Hebiri, Boris Njike, Xavier Siebert (2022) - *Active learning algorithm through the lens of rejection arguments*
4. A Probabilistic Theory of Pattern Recognition (1996) by *Luc Devroye, Laszlo Gyorfi, Gabor Lugosi*
5. A quick overview of Active Learning
6. Active Learning: Curious AI Algorithms
7. Active Learning in Classification — Query Strategies
8. **Dataset :** Stroke Prediction Dataset
9. Gabriel Stoltz (April 15, 2024) An *Introduction to Machine Learning [Lecture Notes]*, Institut Polytechnique de Paris
10. Freund, Y., Seung, H. S., Shamir, E., Tishby, N. (1997). *Selective sampling using the query by committee algorithm*. Machine Learning, 28, 133–168