Stefan Bielmeier Homework 3 - Nov 2nd 2021

Problem 1)

Goal: write $P(Y=1|X) = \frac{1}{1+\exp(-b+w^T X)}$ in terms of

- $\pi = P(Y=1)$ (Bernoulli variable outcome - binary)
- μ_{jk} = mean of Gaussian for $k \in \{0,1\}$
- $X_j \sim \mathcal{N}(\mu_{jk}, \sigma_j)$ for $j = 1, \dots, d$ features
- Cond. independence of $X_1, X_2, X_3 \dots$ given $Y = k$

$\Rightarrow$ Bayes: $P(Y=1|X) = \frac{P(X|Y=1)\cdot P(Y=1)}{P(X)}$ where $P(X) = P(X|Y=1)\cdot P(Y=1) + P(X|Y=0)\cdot P(Y=0)$

$$= \frac{P(X|Y=1)\cdot P(Y=1)}{P(X|Y=1)\cdot P(Y=1) + P(X|Y=0)\cdot P(Y=0)}$$

$$= \frac{1}{1 + \frac{P(X|Y=0)\cdot P(Y=0)}{P(X|Y=1)\cdot P(Y=1)}}$$

with conditional independence of features:

$$P(X|Y=0)\cdot P(Y=0) = P(Y=0)\cdot \prod_{j=1}^{d} P(X_j|Y=0)$$

and $P(X|Y=1)\cdot P(Y=1) = P(Y=1)\cdot \prod_{j=1}^{d} P(X_j|Y=1)$ (chain rule)

$$= \frac{1}{1 + \frac{P(Y=0)}{P(Y=1)}\cdot \frac{\prod_{j=1}^{d} P(X_j|Y=0)}{\prod_{j=1}^{d} P(X_j|Y=1)}}$$

$$= \frac{1}{1 + \left(\frac{1-\pi}{\pi}\right)\cdot \frac{\prod_{j=1}^{d} P(X_j|Y=0)}{\prod_{j=1}^{d} P(X_j|Y=1)}}$$

with $P(X_j|Y=k) = \frac{1}{\sqrt{2\pi\sigma_j^2}} \cdot \exp\left(\frac{-(X_j-\mu_{jk})^2}{2\sigma_j^2}\right)$

Substituting, we get

$$\Rightarrow \frac{\prod_{j=1}^{d} P(X_j|Y=0)}{\prod_{j=1}^{d} P(X_j|Y=1)} = \frac{\prod_{j=1}^{d} \frac{1}{\sqrt{2\pi\sigma_j^2}} \cdot \exp\left(\frac{-(X_j-\mu_{j0})^2}{2\sigma_j^2}\right)}{\prod_{j=1}^{d} \frac{1}{\sqrt{2\pi\sigma_j^2}} \cdot \exp\left(\frac{-(X_j-\mu_{j1})^2}{2\sigma_j^2}\right)}$$

$$= \prod_{j=1}^{d} \exp\left(\frac{-(X_j-\mu_{j0})^2}{2\sigma_j^2} - \frac{-(X_j+\mu_{j1})^2}{2\sigma_j^2}\right)$$

$$= \prod_{j=1}^{d} \exp\left(\frac{-X_j^2 + 2\mu_{j0}X_j - \mu_{j0}^2 + X_j^2 - 2\mu_{j1}X_j + \mu_{j1}^2}{2\sigma_j^2}\right)$$

$$= \prod_{j=1}^{d} \exp\left(\frac{\mu_{j0}-\mu_{j1}}{\sigma_j^2} \cdot X_j + \frac{\mu_{j1}^2 - \mu_{j0}^2}{2\sigma_j^2}\right)$$

$\Rightarrow$ back into other equation:

$$P(Y=1|X) = \frac{1}{1 + \frac{1-\pi}{\pi} \cdot \prod_{j=1}^{d} \exp\left(\frac{\mu_{j0}-\mu_{j1}}{\sigma_j^2} \cdot X_j + \frac{\mu_{j1}^2 - \mu_{j0}^2}{2\sigma_j^2}\right)} \quad \left(\text{we count } \frac{1}{1+\exp(\dots)}\right)$$

$$= \frac{1}{1 + \exp\left(\ln\left(\frac{1-\pi}{\pi} \cdot \prod_{j=1}^{d} \exp(\dots)\right)\right)} = \frac{1}{1 + \exp\left(\ln\left(\frac{1-\pi}{\pi}\right) + \ln \prod_{j=1}^{d} \exp(\dots)\right)}$$

$$= \frac{1}{1 + \exp\left(\ln\left(\frac{1-\pi}{\pi}\right) + \sum_{j=1}^{d} \frac{\mu_{j0}-\mu_{j1}}{\sigma_j^2} \cdot X_j + \frac{\mu_{j1}^2 - \mu_{j0}^2}{2\sigma_j^2}\right)}$$

$$= \frac{1}{1+\exp\left(-\underbrace{\left(-\ln\left(\frac{1-\pi}{\pi}\right) - \sum_{j=1}^{d}\frac{\mu_{j1}^2-\mu_{j0}^2}{2\sigma_j^2}\right)}_{b} + \underbrace{\sum_{j=1}^{d}\frac{\mu_{j0}-\mu_{j1}}{\sigma^2}}_{\omega_j}\cdot X_j\right)}$$

HW 3 ctd
SB
Nov 2nd

"Vector" ω_j X_j !

$$\hat{=} \frac{1}{1+\exp\left(-b+\omega^T \cdot X\right)} \quad \checkmark\checkmark$$

Problem 2)

a) i) Simple case of X_1 and X_2

$\Rightarrow Var(X_1+X_2) = Var(X_1) + 2Cov(X_1,X_2) + Var(X_2)$

knowing that $Cov(X_1,X_2)=0$ because they're independent variables, $Var(X_1+X_2) = Var(X_1) + Var(X_2) - \sum_{i=1}^{2} Var(X_i)$

$$\Rightarrow Var\left(\underbrace{\frac{\sum_{i=1}^{N} X_i}{n}}_{\text{Average}}\right) = \frac{1}{n^2} Var\left(\sum_{i=1}^{N} X_i\right) = \frac{1}{n^2}\sum_{i=1}^{n} Var(X_i) \quad (\text{all } X_i\text{s indep})$$

with $Var(cX) = c^2 Var(X)$

$$= \frac{1}{n^2}\sum_{i=1}^{n}\sigma_i^2 \quad \text{, with all variances the same: } \sigma_i^2 = \sigma^2$$

$$= \frac{1}{n^2}\cdot n\cdot\sigma^2 = \frac{\sigma^2}{n}$$

ii) Simple: $P(\text{"data point not chosen in 1 draw"}) = 1-\frac{1}{M}$ (Single random draw from M)

Drawing M times: $P(OOB) = \left(1-\frac{1}{M}\right)^M$

for large M: $\lim_{M\to\infty}\left(1-\frac{1}{M}\right)^M = \frac{1}{e} \approx 0.37$ $\checkmark\checkmark$

a iii) The OOB error works the following way:

1) generate B samples of length n of dataset with length n.
2) Train a model on each sample B_1, B_2, etc.
3) Deliberately make models do a prediction on an observation that they didn't include in training sample.
4) Average that prediction, and compare to real value ≙ OOB error for single observation
5) Do for all observations
   ↳ OOB error = $\frac{\text{Falsely pred. observations (OOB)}}{\text{all observations (OOB)}}$

→ The OOB error decreases for a given model with the more trees it has ⇒ is better able to classify observations it "didn't know".

→ At some point, increasing the # of trees will become computionally more expensive, while the OOB error only decreases slightly / marginally less

→ Choose the # of trees when OOB isn't decreasing significantly anymore.

2a iv)

SB

Nov 3rd 2021

- Var for correlated random vars $X_1, X_2, \dots, X_n$

$$\Rightarrow \operatorname{Var}\left(\frac{\sum_{i=1}^{n} X_i}{n}\right) = \frac{1}{n^2} \cdot \operatorname{Var}\left(\sum_{i=1}^{n} X_i\right)$$

$$= \frac{1}{n^2}\left(\sum_{i=1}^{n} \operatorname{Var}(X_i) + \sum_{i=1}^{n}\sum_{i \neq j} \operatorname{Cov}(X_i, X_j)\right)$$

upper bound: $\leq \frac{1}{n^2} \cdot \left(n \cdot \sigma_i^2 + n \cdot (n-1) \cdot \delta^2\right)$ (with $\operatorname{Cov}(X_i, X_j) \leq \delta^2$)

$\leq \frac{1}{n^2} \cdot \left(n \cdot \sigma^2 + (n^2 - n)\delta^2\right)$ with $\sigma_i^2 = \sigma^2$

$\leq \frac{1}{n^2} \cdot \left(n\sigma^2 + (n^2 - n)\delta^2\right)$

$\leq \frac{\sigma^2}{n} + \frac{n^2\delta^2 - n\delta^2}{n^2} = \frac{\sigma^2}{n} + \delta^2 - \frac{\delta^2}{n}$

$\leq \underline{\frac{1}{n} \cdot \sigma^2 + \left(1 - \frac{1}{n}\right) \cdot \delta^2}$

v) with $N \to \infty$:

$$\lim_{n \to \infty} \underbrace{\frac{1}{n}}_{\to 0} \sigma^2 + \underbrace{\left(1 - \frac{1}{n}\right)}_{\to 0} \cdot \delta^2 = 0 + (1 - 0)\delta^2 = \delta^2$$

from part i: $\lim_{n \to \infty} \frac{\sigma^2}{n} = 0$

$\Rightarrow$ Bagging is limited by the covariance of the dataset's features or random variables (which we see in the trees' covariance).

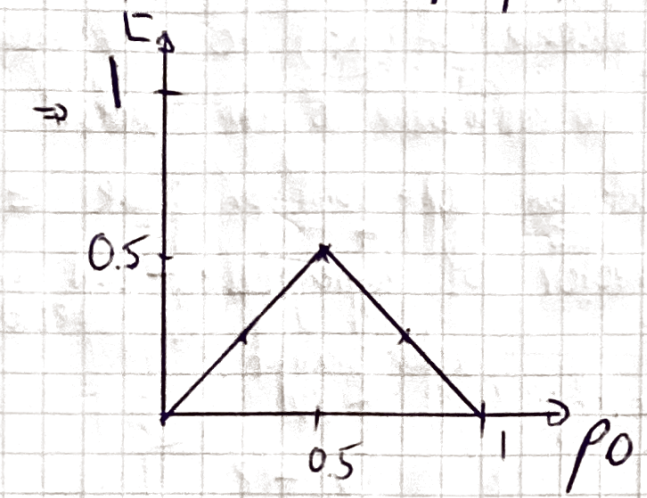
$\Rightarrow$ Bagging is limited by the degree of correlation between the trees.

SB
Nov 3rd
2021

2b) $E = 1 - \max_k p_k$ with $k \in \{0,1\}$

$= 1 - \max(p_1, p_0)$ with $p_0 = 1 - p_1$

(classification error rate)



! we use p_0 as first class ! (doesn't matter for plots)

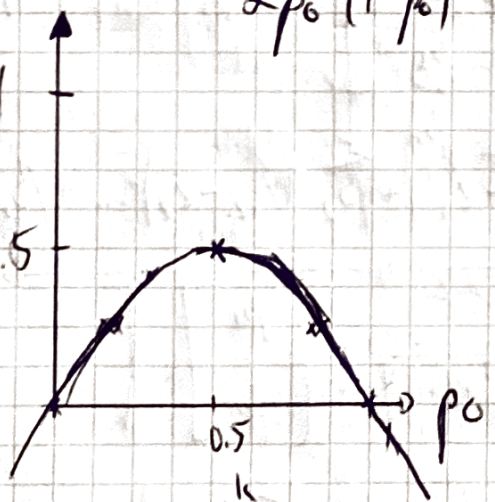
Gini-Index:

$\sum_{k=1} p_k(1-p_k)$, starting at $k=0$ with $k \in \{0,1\}$

$\Rightarrow \sum_{k=0} p_k(1-p_k) = p_0(1-p_0) + p_1(1-p_1) = p_0 \cdot p_1 + p_1 p_0 = 2p_1 p_0$

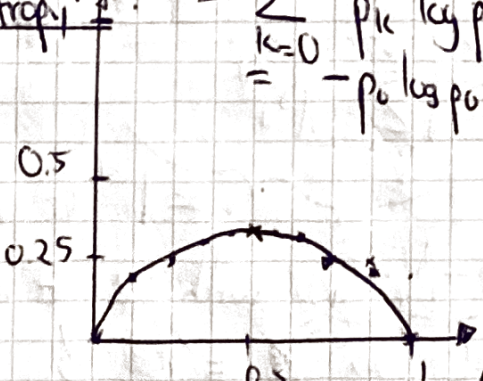
$= 2p_0 \cdot (1-p_0) = -2p^2 + 2p_0$

$\Rightarrow$



Entropy: $-\sum_{k=0}^{k} p_k \log p_k = -(p_0 \log p_0 + p_1 \log p_1)$

$= -p_0 \log p_0 - (1-p_0)\log(1-p_0)$



Problem 3a)

Hr 3rd SB

we want to calculate the distance of x_i to hyperplane (perpendicular distance). That is equivalent to the scalar projection of b (vector between same point on plane z and x_i) on n (normal vector of hyperplane - unit normal vector)

$$d = |\text{comp}_n b| = \frac{|b \circ n|}{|n|} \quad \text{with: } \circ \text{ scalar product.}$$

$$x_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \dots \\ x_{in} \end{bmatrix} \quad z = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ \vdots \\ z_n \end{bmatrix} \Rightarrow b = \begin{bmatrix} x_{i1} - z_1 \\ x_{i2} - z_2 \\ \dots \\ x_{in} - z_n \end{bmatrix}$$

and $n = \frac{\langle w \rangle}{|w|}$ with $\langle w \rangle$ is $\beta_1, \beta_2 \dots, \beta_n$ (weights of hyperplane)

$$= \frac{1}{\sqrt{\beta_1^2 + \beta_2^2 + \dots + \beta_n^2}} \cdot \begin{bmatrix} \beta_1 \\ \beta_2 \\ \dots \\ \beta_n \end{bmatrix}$$

$$\Rightarrow d = \frac{\left| \frac{1}{\sqrt{\beta_1^2 + \beta_2^2 + \dots + \beta_n^2}} \cdot \left(\beta_1 \cdot (x_{i1} - z_1) + \beta_2 \cdot (x_{i2} - z_2) + \dots + \beta_n \cdot (x_{in} - z_n) \right) \right|}{\sqrt{\frac{1}{\sqrt{\beta_1^2 + \beta_2^2 + \dots + \beta_n^2}} \cdot \begin{bmatrix} \beta_1 \\ \dots \\ \beta_n \end{bmatrix}}}$$

with $\left| \begin{bmatrix} \beta_1 \\ \dots \\ \beta_n \end{bmatrix} \right| = \sqrt{\sum_{j=1}^{p} \beta_j^2}$

$$:= \frac{|\beta_1 \cdot x_{i1} + \beta_2 \cdot x_{i2} + \dots \beta_n \cdot x_{in} - \beta_1 \cdot z_1 - \beta_2 \cdot z_2 - \dots - \beta_n \cdot z_n|}{\sqrt{\sum_{j=1}^{p} \beta_j^2}}$$

$\sqrt{\sum_{j=1}^{p} \beta_j^2} = 1$ (constraint)

and with $(-\beta_1 \cdot z_1 - \beta_2 \cdot z_2 - \dots - \beta_n \cdot z_n)$ with $z \in$ hyperplane

$\Rightarrow \beta_1 \cdot z_1 + \beta_2 \cdot z_2 + \dots + \beta_n \cdot z_n + \beta_0 \overset{!}{=} 0 \Rightarrow = \beta_0$

$$\Rightarrow d = |\beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_n x_{in} + \beta_c|$$
$$= |\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}|$$

now: max margin classifier: Classify based on sign of $f(x_i)$ with $f(x_i) = \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}$

$\Leftrightarrow$ $\text{sign}(y_i) = \text{sign}(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij})$

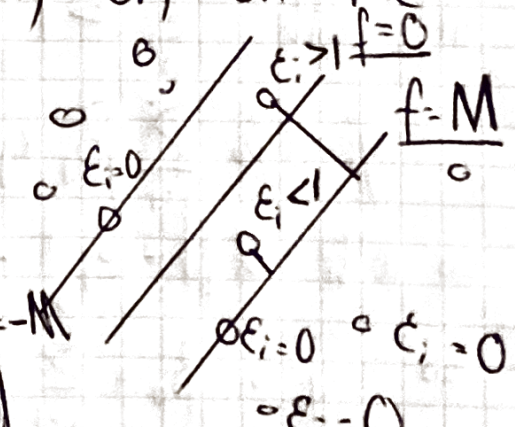
$\Rightarrow d = y_i \cdot (\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}) = |\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}|$

$(-1 \cdot -1 = 1,\ 1 \cdot 1 = 1)$ ✓

Problem 3b)

(i) given point x_i, with $\varepsilon_i = 0$

$d \geq M \cdot (1 - \varepsilon_i) \Rightarrow d \geq M$ x_i is on the correct side of margin ✓ (on either side) or, on the margin support vector!, at least $\geq M$ away from hyperplane



as x_i has a distance d to hyperplane, that is $\geq M$, we can safely assume that it's classified correctly $\Rightarrow$ On correct side of hyperplane. ✓

We know that the optimization problem will fulfill this constraint $\text{sign}(y_i) \cdot \text{sign}(d_{raw}) \geq M$ $\Rightarrow$ needs to be equal sign $\Rightarrow$ correct class!

3b ii)

$0 < \varepsilon_i < 1$ for point x_i

⇒ is it on the correct side of the hyperplane?

↳ $y_i \left(\beta_0 + \sum_{j=1}^{p} \beta_j \cdot x_{ij}\right) \geq M \cdot (1 - \varepsilon_i)$ with $0 < \varepsilon_i < 1$

⇒ $0 < y_i \left(\beta_0 + \sum_{j=1}^{p} \beta_j \cdot x_{ij}\right) < M$

⇒ point x_i is classified correctly, (signs of y_i must match $\beta_0 + \sum_{j=1}^{p} \beta_j \cdot x_{ij}$ to satisfy inequality above)
(not on the hyperplane as $d > 0$)
however, a loss incurred because $\varepsilon_i > 0$.

The point x_i is not on the correct side of the margin, but between the hyperplane and the margin support vector,

with $0 < d(x_i) < M$

iii) with $\varepsilon_i > 1$ for x_i (one point)

$y_i \cdot \left(\beta_0 + \sum_{j=1}^{p} \beta_j \cdot x_{ij}\right) \geq M \cdot (1 - \varepsilon_i)$

⇒ with $M \cdot (1 - \varepsilon_i) < 0$ for $\varepsilon_i > 1$

⇒ $y_i \cdot \left(\beta_0 + \sum_{j=1}^{p} \beta_j \cdot x_{ij}\right) < 0$ (has a negative sign)

⇒ Misclassified, as $\text{sign}(y_i) \neq \text{sign}\left(\beta_0 + \sum_{j=1}^{p} \beta_j \cdot x_{ij}\right)$

⇒ Not on correct side of hyperplane,

⇒ Not on correct side of the margin!

SB
3rd Nov 2021

3c)

Show that $f(x^*) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x^*, x_i \rangle$

is equal to $f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \dots + \beta_p \cdot x_p^*$,

with: $\mathbb{R}^p$, and x_i^{th} notes the i^{th} training observation.

⊕ ⇒ Find values for β_j, $j \in \{1, \dots, p\}$ in terms of α_i & x_i for $i \in \{1, \dots, n\}$

$$\Rightarrow f(x^*) = \beta_0 + \sum_{i=1}^{n} \alpha_i \cdot \langle x^*, x_i \rangle \quad \text{with } \langle x^*, x_i \rangle = (x_1^* \cdot x_{i1} + x_2^* \cdot x_{i2} + \dots + x_p^* \cdot x_{ip}) \text{ (Scalar product)}$$

$$= \beta_0 + \sum_{i=1}^{n} \alpha_i \left(x_1^* \cdot x_{i1} + x_2^* \cdot x_{i2} + \dots + x_p^* \cdot x_{ip} \right)$$

$$= \beta_0 + \sum_{i=1}^{n} \alpha_i \cdot \sum_{j=1}^{p} x_j^* \cdot x_{ij} \quad \text{(we want } \textstyle\sum_{j=1}^{p} \text{ outside)}$$

$$= \beta_0 + \sum_{i=1}^{n} \sum_{j=1}^{p} \alpha_i x_j^* \cdot x_{ij} = \beta_0 + \sum_{j=1}^{p} x_j^* \sum_{i=1}^{n} \alpha_i \cdot x_{ij}$$

$$= \beta_0 + x_1^* \cdot \sum_{i=1}^{n} \alpha_i \cdot x_{i1} + x_2^* \cdot \sum_{i=1}^{n} \alpha_i x_{i2} + \dots + x_p^* \cdot \sum_{i=1}^{n} \alpha_i x_{ip}$$

$$\Rightarrow \text{with } \underline{\beta_j = \sum_{i=1}^{n} \alpha_i \cdot x_{ij}} \text{ for all } j \in \{1, \dots, p\}$$

$$\hat{=} \beta_0 + x_1^* \cdot \beta_1 + \dots + x_p^* \cdot \beta_p \quad \checkmark\checkmark$$

Oct 21st 2021

CS 194-172 – Computational Genomics – HW 3

Stefan Bielmeier

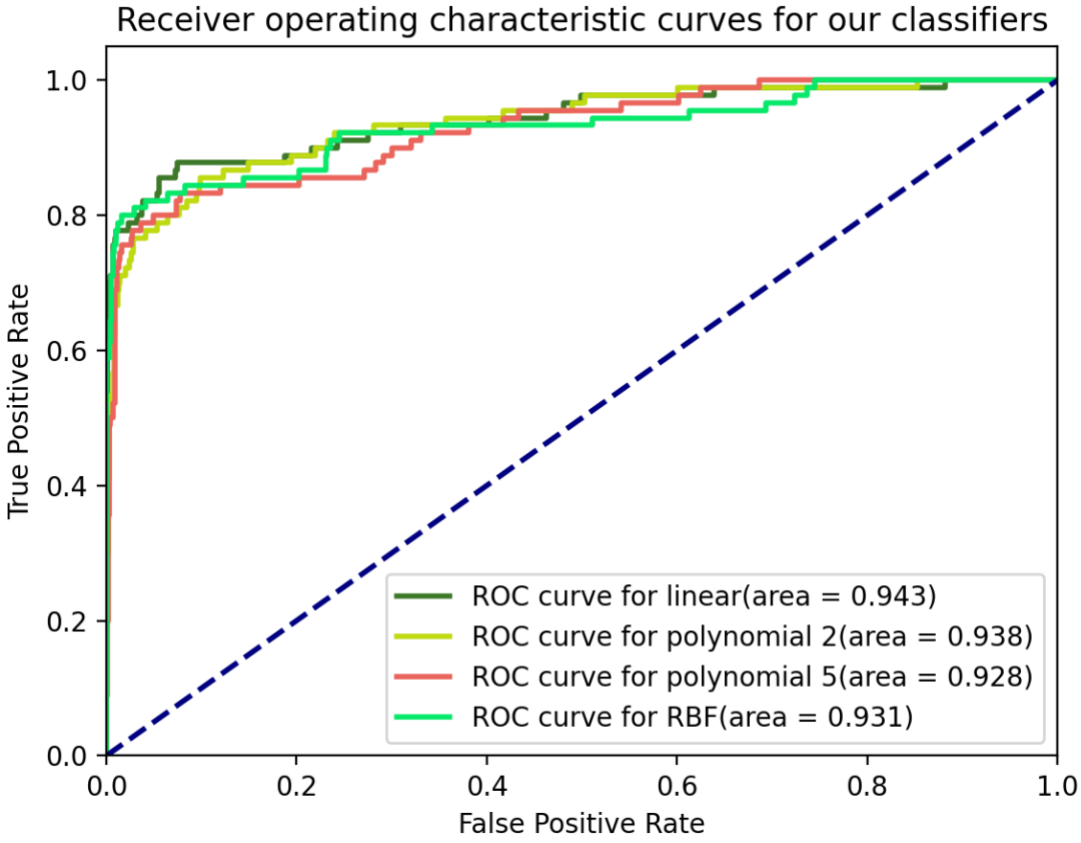
Problem 4a) and b)

SVM Kernel Type	**Best value of C**	**Mean cross-validation accuracy for that C**	**Test accuracy for that C**	**Total running time (secs) of cross-validation for all possible Cs** (0.1, 1, 10, 100, 1000)
Linear	0.1	0.981	0.974	306.0
Polynomial degree 2	0.1	0.981	0.953	274.9
Polynomial degree 5	1000	0.973	0.963	0.4
RBF Kernel	1000	0.98	0.97	< 0.0

Raw data

```
acc for poly2 for C=0.1 0.9810000000000001
acc for poly2 for C=1.0 0.9800000000000001
acc for poly2 for C=10.0 0.9790000000000001
acc for poly2 for C=100.0 0.9800000000000001
acc for poly2 for C=1000.0 0.976
cross-validation took 274.9 secs
acc for poly5 for C=0.1 0.913
acc for poly5 for C=1.0 0.9400000000000001
acc for poly5 for C=10.0 0.968
acc for poly5 for C=100.0 0.9709999999999999
acc for poly5 for C=1000.0 0.9730000000000001
cross-validation took 0.4 secs
acc for rbf for C=0.1 0.945
acc for rbf for C=1.0 0.97
acc for rbf for C=10.0 0.975
acc for rbf for C=100.0 0.9799999999999999
acc for rbf for C=1000.0 0.9800000000000001
cross-validation took 0.0 secs
acc for linear for C=0.1 0.9810000000000001
acc for linear for C=1.0 0.9800000000000001
acc for linear for C=10.0 0.9790000000000001
 acc for linear for C=100.0 0.9800000000000001
acc for linear for C=1000.0 0.976
cross-validation took 306.0 secs
```

Problem 4c)
ROC curves for each classifier and respective AUROC in label as “area”.



Problem 4d)

The linear Kernel probably trains the longest because it tries to separate the data into zero and ones with the most constrained assumption of separation, a single Hyperplane. Finding a hyperplane of dimension *K – 1*, where k is the *number of columns*, that almost linearly separates the data into two classes – zero (0) and one (1) probably needs quite the number of iterations, because it is quite unlikely that the data is linearly separable at all, the bigger K gets for the same number of observations N.

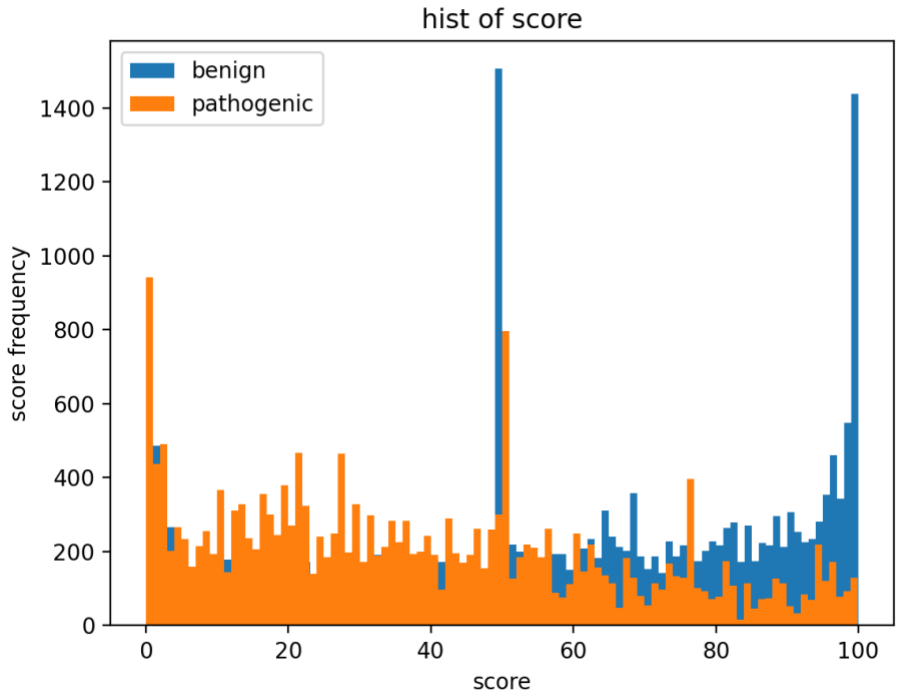
In contrast, a polynomial of 2nd degree SVM may converge quicker as it basically is not constrained to one, but **two** Hyperplanes / Support vectors that separate the data. That gives more flexibility to train the model and may lead it to converge quicker (find a local optimum) to appropriate parameters. A polynomial of 5th degree SVM has 5 possible support vectors to separate the data into 0s and 1s, given that K dimensions and the number of observations stays the same.
This is also why we observe inversely correlated model training times with dimensionality of the model.

Problem 5a)
After this filtering, 24346 benign and 20596 pathogenic variants remain.

Problem 5b)
After randomly splitting into 80% training data, and 20% validation data, there are

- 16517 pathogenic and 19436 benign in the training set
- 4079 pathogenic and 4910 benign variants in the validation set

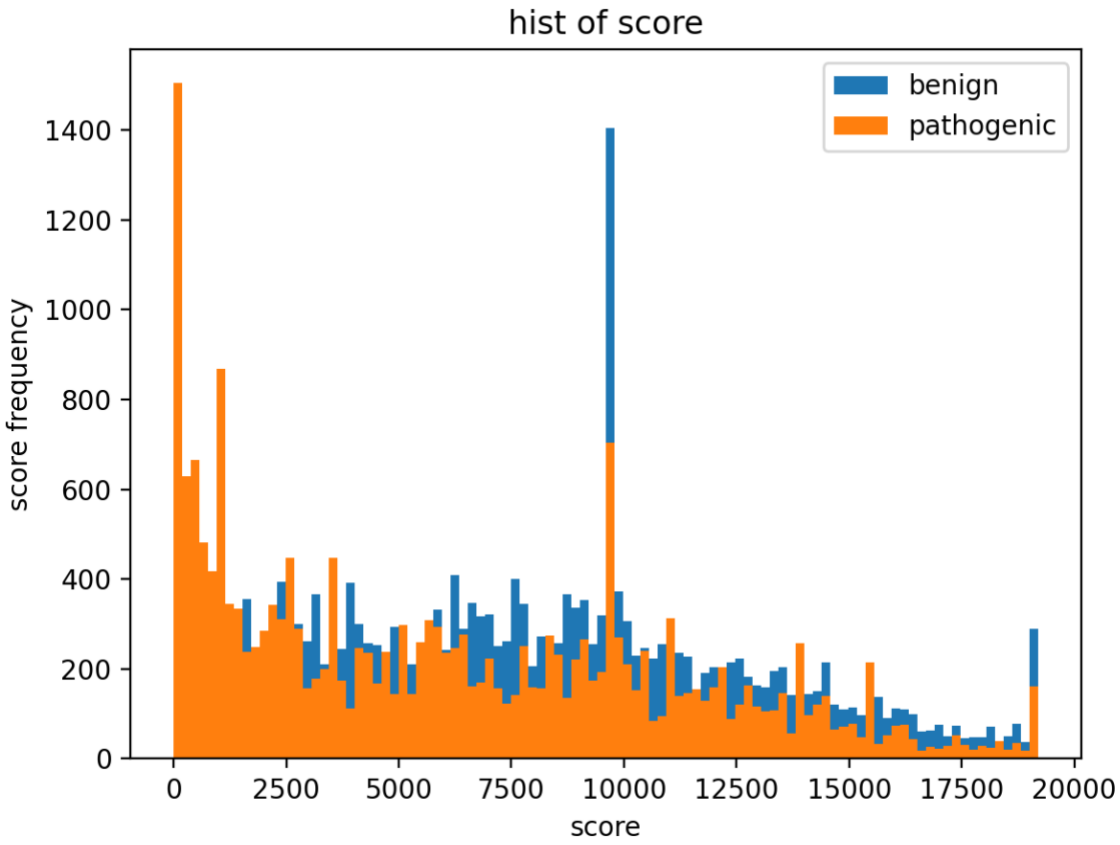
Problem 5c)



Based on the plot, do you think RVIS is a good feature to add to our model?

A gene with a positive score has more common functional variation, and a gene with a negative score has less and is referred to as "intolerant". The percentiles work the same way, the higher, the more common, the lower, the less common. Therefore, RVIS seems like a good feature to add to the model because it can help the model distinguish more common function variants (benign) from less common functional variants (potentially pathogenic).

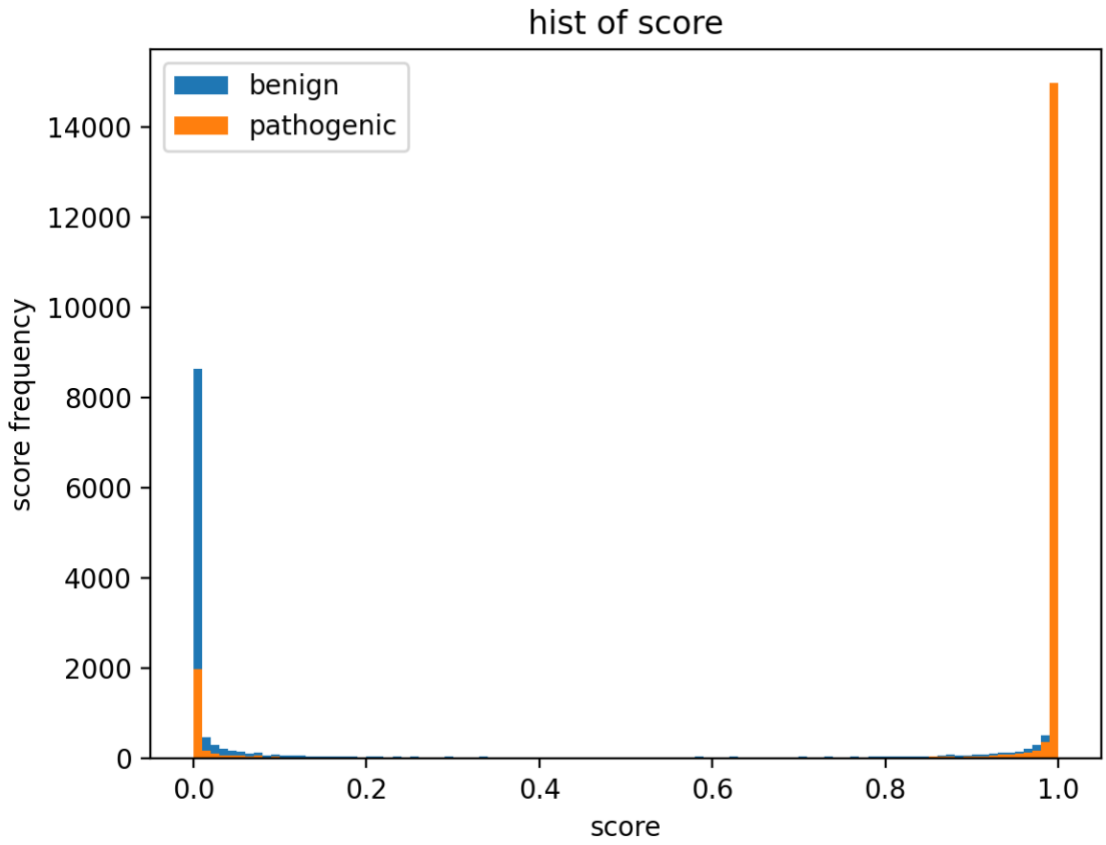
Problem 5d)



This seems like an okay metric. Very high ranks (i.e. number 1, 2) usually are only assigned to pathogenic variants, while high to very low ranks (1000 – 20000, etc.) are assigned to benign or pathogenic variants in a seemingly similar distribution.

Usually high ranks are associated more with pathogenic scores than benign, any other rank says: doesn't matter, which may help our model predict as well.

Problem 5e)



The phastCons score for each nucleotide describes the probability that the nucleotide is part of a conserved region.

- ⇨ High phastCons score = highly conserved = a mutation in highly conserved (i.e. important) regions is probably damaging
- ⇨ Low phastCons score = not conserved at all = probably not so damaging if a mutation occurs but can also be damaging. That's why we see both pathogenic and benign variants, but way more benign ones.

I think this is an OK feature, as it does not separate the pathogenic or benign scores well-enough so that the machine learning model can make clear distinctions because of low scores showing both categories. However, a high score very clearly correlates with pathogenicity.

Problem 5f)

I chose a Random-Forest Classifier. I trained the model on 60% training data to achieve a better accuracy. I used the data compiler Brainome (CS 294-082, brainome.ai). I did not need to perform cross-validation, as the data compiler trains the model with training data that approximately resembles the split of classes in the entire dataset. In our case, it's about 54.17% benign variants, and 45.83% pathogenic variants. This is why I used a split into 60% training data to achieve best results for training accuracy and validation accuracy (combined model accuracy: 85.57%).

Validation accuracy on the 40% test set: 79.35%

The chosen hyperparameters for the Random Forest are:

- n_estimators = number of trees in the foreset
- max_features = max number of features considered for splitting a node
- max_depth = max number of levels in each decision tree
- min_samples_split = min number of data points placed in a node before the node is split
- min_samples_leaf = min number of data points allowed in a leaf node
- bootstrap = method for sampling data points (with or without replacement)

When I split into a randomly selected 20% validation set. My model achieves the following ROC curve on that randomly selected 20% validation set, and an AUROC of 0.933.