Mid (Fall 22) Machine Learning

Q1) True / False

1. Unsupervised Learning enables an agent to learn in an interactive environment through trial and error using the feedback of its own actions and experiences. False

2. Cross-validation splits the dataset into k-partitions (folds) and is preferable only for large datasets. False

3. Gradient descent is an optimization algorithm used to find the values of parameters of a function that minimizes a cost function. True

4. Logistic regression is an example of supervised learning, and it is used to calculate or predict the probability of a binary (yes/no) event (class) occurring. True

5. All learning algorithms require transforming labels (categorical feature) into numbers. False

6. Just removing data examples with missing features from a dataset could be an effective way to deal with missing values sometimes. True

7. Binning (also called bucketing) is the process of converting a continuous feature into multiple binary features called bins or buckets, typically based on value range. True

8. An underfitting problem happens when a model predicts very well the training data but poorly the data from a holdout set (e.g., testing set). False

9. Accuracy is not a useful metric when errors in predicting all classes are equally important. False

10. Model-based learning algorithms use the whole dataset as the model such as k-Nearest Neighbors (kNN). False

Q2)

1. How Logistic Regression is different from Linear Regression? Name two differences between them.
   - Linear Regression used to handle regression problem, whereas Logistic Regression handle classification problem
   - Linear Regression output is continuous whereas Logistic Regression discrete output (between 0 and 1)

2. Name a significant disadvantage of Decision Tree models.
   - It generally leads to overfitting of the data which ultimately leads to wrong predictions.
   - Affected by noise.

3. What is the purpose of the kernel function in SVM?
   - It can make the decision boundary arbitrarily non-linear "transforms the training set of data so that a non-linear decision surface can transform to a linear equation in a higher number of dimension spaces." Briefly: The kernel function separates the data by adding dimensions to the problem. Like (RBF) Kernel

4. How does the k-Nearest Neighbors (kNN) algorithm calculate the distance between the data points?
   - KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression). Euclidean distance is frequently used in practice to calculate the distance.

5. Show how this dataset is transformed by the One-Hot encoding process.

Car_Brand	Car_Color
Toyota	White
Nissan	Black
Ford	White
Nissan	Red

So, for One-Hot encoding each feature will be in separate column.

Toyota	Nissan	Ford	White	Black	Red
1	0	0	1	0	0
0	1	0	0	1	0
0	0	1	1	0	0
0	1	0	0	0	1

Q3)

Suppose that we have built a model to classify student performance in a course into the following classes: Poor; Average, and Excellent. After evaluating the model on a test set, we got the following confusion matrix:

		Predicted		
		Poor	Average	Excellent
Actual	Poor	10	4	1
	Average	2	34	6
	Excellent	2	5	36

1. Calculate the accuracy of the classifier.

$$\frac{10+34+36}{10+4+1+2+34+6+2+5+36} = 0.8$$

2. Calculate the precision of the "Poor" class.

$$\frac{10}{10+2+2} = 0.71$$

3. Calculate the recall of the "Excellent" class.

$$\frac{36}{36+5+2} = 0.83$$

4. Is the test set balanced or not? Justify your answer.

   15 (Poor), 42(Average), 43(Excellent) >> It's imbalanced since the classes not equally distributed.

Out Of The Exam

Please note that I took these points from book I didn't write them, but I picked them.

Q4)

Name a difference between loss and cost function?

- The loss function is to capture the difference between the actual and predicted values (error) for a single record whereas cost functions (empirical risk) aggregate the difference for the entire training dataset.

What's the difference between Model-Based and Instance-Based algorithms?

- Model-based learning algorithms use the training data to create a model that has parameters learned from the training data. After the model was built, the training data can be discarded. Like SVM
- Instance-based learning algorithms use the whole dataset as the model. Like kNN

Determine the difference between Parameters and Hyperparameters?

- A hyperparameter is a property of a learning algorithm, usually (but not always) having a numerical value.

- Parameters are variables that define the model learned by the learning algorithm. Parameters are directly modified by the learning algorithm based on the training data.

Determine the difference between Shallow and Deep Learning?

- A shallow learning algorithm learns the parameters of the model directly from the features of the training examples.
- deep learning, most model parameters are learned not directly from the features of the training examples, but from the outputs of the preceding layers.

Explain difference between equations for Linear regression and SVM

- Difference is the missing sign operator. And Also the hyperplane in SVM plays the role of the decision boundary (To separate two groups of example) but in linear regression (To be as close to all training examples as possible)

- Plane and Hyperplane for
- $(\mathrm{fw}, \mathrm{b}(\mathrm{xi}) - \mathrm{yi})^2$ squared error loss.
- average squared error loss, also known as the mean squared error or MSE.
- binary loss (1 when f(xi) and yi are different and 0 when they are the same)
- Overfitting is the property of a model such that the model predicts very well labels of the examples used during training.
- One practical justification of the choice of the linear form for the model is that it's simple. Why use a complex model when you can use a simple one? Another consideration is that linear models rarely overfit.
- linear regression can be useful? it doesn't overfit much.
- The absolute value is not convenient? because it doesn't have a continuous derivative, which makes the function not smooth.
- Gradient descent is a method of updating w and b to reduce (minimize) the cost function (for example, MSE).
- logistic regression it can naturally be extended to multiclass classification.
- standard logistic function (also known as the sigmoid function)

- The optimization criterion in logistic regression is called **maximum likelihood** Instead of minimizing the average loss, like in linear regression, we maximize the likelihood.
- logistic regression builds parametric model, the ID3 builds nonparametric model (Non-parametric Models are statistical models that do not often conform to a normal distribution)
- In ID3, the goodness of a split is estimated by using the criterion called **entropy.**
- **Entropy** reaches its maximum when all values of the random variables are equiprobable. Entropy reaches its minimum when the random variable can have only one value.
- **C4.5** handles incomplete examples.
- **C4.5** solves overfitting problem by using a bottom-up technique known as "pruning".

  Explain Pruning?

  - consists of going back through the tree once it's been created and removing branches that don't contribute significantly enough to the error reduction by replacing them with leaf nodes.

- The entropy-based split criterion intuitively makes sense: entropy reaches its minimum of 0 when all examples in S have the same label; on the other hand, the entropy is at its maximum of 1. when exactly one-half of examples in S is labeled with 1. (مكرر I know)
- **Hing loss** for dealing with noise in SVM.
- SVMs that optimize hinge loss are called **soft-margin** SVMs, while the original formulation is referred to as a **hard-margin** SVM.
- In SVM C determines the tradeoff between increasing the size of the decision boundary and ensuring that each xi lies on the correct side of the decision boundary. As we decrease the value of C, making classification errors is becoming more costly. a larger margin is better for generalization.
- C regulates the tradeoff between classifying the training data well (minimizing empirical risk)

- k-Nearest Neighbors (kNN) is a non-parametric learning algorithm(since the number of parameters grows with the size of the training set.).
- kNN keeps all training examples in memory.
- kNN take majority label, in case of classification, or the average label, in case of regression.
- distance functions like (Euclidean, cosine similarity, Chebychev, Mahalanobis, and Hamming)
- **Gradient descent** is an iterative optimization algorithm for finding the minimum of a function.
- **Gradient descent** can be used to find optimal parameters for linear and logistic regression, SVM and neural network.
- Convex functions have only one minimum, which is global. Optimization criteria for neural networks are not convex.
- Gradient descent starts with **calculating** the partial derivative for every parameter.
- Gradient descent proceeds in **epochs**. An epoch consists of using the training set entirely to update each parameter. Every repetition called epoch.
- The learning rate **α** controls the size of an update.
- We subtract (as opposed to adding) partial derivatives from the values of parameters because derivatives are indicators of growth of a function.
- Typically, we need many epochs until we start seeing that the values for w and b don’t change much after each epoch; then we stop.
- SGD (stochastic gradient descent) itself has various “upgrades”. **Adagrad** is a version of SGD that scales α for each parameter according to the history of gradients. and vice-versa. **Momentum** is a method that helps accelerate SGD
- **gradient descent and its variants** are <u>not</u> machine learning algorithms. They are solvers of minimization problems in which the function to minimize has a gradient.
- **decision tree** learning, can accept categorical features.
- SVM, logistic and linear regression, as well as kNN (with cosine similarity or Euclidean distance metrics), expect (need) numerical values for all features.
- All algorithms implemented in scikit-learn expect numerical features.

- Some classification algorithms (like decision tree learning, logistic regression, or SVM) build the model using the whole dataset at once. If you have got additional labeled examples, **you have** to rebuild the model from scratch.
- Other algorithms (such as Naïve Bayes, multilayer percep-tron (MLP), SGDClassifier/SGDRegressor, PassiveAggressiveClassifier/PassiveAggressiveRegressor in scikit-learn) can be trained iteratively. you can update the model using only the new data.
- algorithms, like decision tree learning, SVM, and kNN can be used for both **classification and regression** while others can only solve one type of problem.
- The problem of transforming raw data into a dataset is called **feature engineering.**
- A **feature vector** is a vector in which each dimension j = 1, . . ., D contains a value that describes the example somehow.
- The role of the data analyst is to create **informative features.**
- **Highly informative features** are also called features with **high predictive power.**
- We say that a model has a **low bias** when it predicts the training data well.
- **You should not** transform red into 1, yellow into 2, and green into 3 to avoid increasing the dimensionality because that would imply that there's an order and which may potentially lead to overfitting.
- is when you have a numerical feature, but you want to convert it into a categorical one. **Binning** (also called bucketing) is the process of converting a continuous feature into multiple binary features called bins. typically based on value range.

Explain Binning?

- For example, instead of representing age as a single real-valued feature. the analyst could chop ranges of age into discrete bins: all ages between 0 and 5 years-old could be put into one bin, 6 to 10 years-old could be in the second bin, 11 to 15 years-old could be in the third bin, and so on.

- **Normalization** is the process of converting an actual range of values which a numerical feature can take, into a standard range of values, typically in the interval [−1, 1] or [0, 1].

More generally, the normalization formula looks like this:

$$\bar{x}^{(j)} = \frac{x^{(j)} - min^{(j)}}{max^{(j)} - min^{(j)}},$$

Figure 1 THISSS important?

Explain Normalization?

- For example, suppose the natural range of a particular feature is 350 to 1450. By subtracting 350 from every value of the feature, and dividing the result by 1100, one can normalize those values into the range [0, 1].

- Normalizing the data is not a strict requirement. However, in practice, it can lead to an **increased speed of learning**.
- Normalization avoid problem with dealing with small or large number (known as **numerical overflow**)
- **Standardization** (or z-score normalization) is the procedure during which the feature values are rescaled.

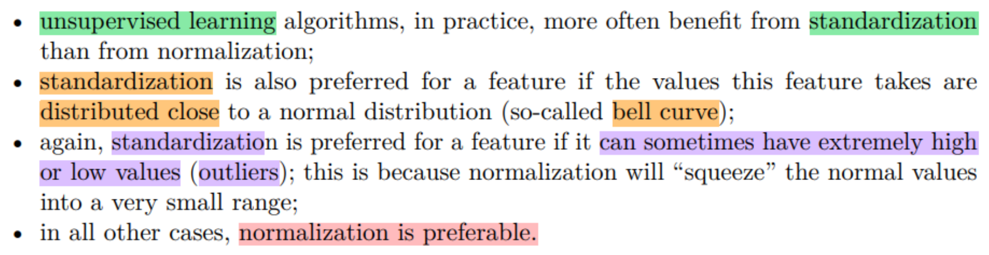
$\mu = 0$ and $\sigma = 1$, where μ is the mean (the average value of the feature, averaged over all examples in the dataset) and σ is the standard deviation from the mean.

Standard scores (or z-scores) of features are calculated as follows:

$$\hat{x}^{(j)} = \frac{x^{(j)} - \mu^{(j)}}{\sigma^{(j)}}.$$

- You may ask when you should use **normalization** and when **standardization**? There's no definitive answer to this question. Usually, if your dataset is not too big and you have time, you can try both and see which one performs better for your task.

If you don't have time to run multiple experiments, as a rule of thumb:



- unsupervised learning algorithms, in practice, more often benefit from standardization than from normalization;
- standardization is also preferred for a feature if the values this feature takes are distributed close to a normal distribution (so-called bell curve);
- again, standardization is preferred for a feature if it can sometimes have extremely high or low values (outliers); this is because normalization will "squeeze" the normal values into a very small range;
- in all other cases, normalization is preferable.

How to deal with Missing values?

- 1) removing the examples
- 2) using a learning algorithm that can deal with missing feature values
- 3) using a data imputation technique

Determine techniques for data imputation?

- One data imputation technique consists in replacing the missing value of a feature by an average value.
- Another technique is to replace the missing value with a value outside the normal range of values. For example, if the normal range is [0, 1], then you can set the missing value to 2.
- you can replace the missing value by a value in the middle of the range. For example, if the range for a feature is [−1, 1], you can replace the missing value by a value in the middle of the range. For example, if the range for a feature is [−1, 1], you can set the missing value to be equal to 0.
- A more advanced technique is to use the missing value as the target variable for a regression problem.
- Finally, if you have a significantly large dataset and just a few features with missing values, you can increase the dimensionality of your feature vectors by adding a binary indicator feature for each feature with missing values.

Explain how increase dimensionality help to deal with missing values?

- Let's say feature j = 12 in your D-dimensional dataset has missing values. For each feature vector x, you then add the feature j = D + 1 which is equal to 1 if the value of feature 12 is present in x and 0 otherwise. The missing feature value then can be replaced by 0 or any number of your choice.

What are the criteria for selecting learning algorithms?

- 1) Explainability

(Does your model have to be explainable to a non-technical audience) Most very accurate learning algorithms are so-called "black boxes." Examples of such models are neural networks or ensemble models. **<u>On other hand</u>** kNN, linear regression, or decision tree the way they make their prediction is very straightforward.

- 2) In-memory vs. out-of-memory

If the dataset be fully loaded into the RAM of your server or personal computer, you can choose from a wide variety of algorithms. Otherwise, you would prefer **incremental learning algorithms (adding more data gradually).**

- 3) Number of features and examples

neural networks and gradient boosting (we consider both later), can handle a huge number of examples and millions of features. Others, like SVM, can be very modest in their capacity.

- 4) Categorical vs. numerical features
- 5) Nonlinearity of the data

Is your data linearly separable or can it be modeled using a linear model? If yes, SVM with the linear kernel, logistic or linear regression can be good choices. Otherwise, deep neural networks or ensemble algorithms, discussed in Chapters 6 and 7, might work better.

- 5) Training speed

Neural networks are known to be slow to train. Simple algorithms like logistic and linear regression or decision trees are much faster. Some like random forests, benefit from the availability of multiple CPU cores

- 6) Prediction speed

kNN, ensemble algorithms, and very deep or recurrent neural networks, are slower.

Determine the three distinct sets of labeled examples?

- 1) training set 2) validation set 3) test set.

The validation and test sets are roughly the same sizes. The learning algorithm cannot use examples from these two subsets to build the model. That is why those two sets are often called **holdout sets**.

Why do we need two holdout sets and not one?

- We use the validation set to 1) choose the learning algorithm and 2) find the best values of **hyperparameters**. We use the test set to assess.

- **low bias** if it predicts well the labels of the training data.
- If the model makes many mistakes on the training data, we say that the model has a **high bias.**
- **underfitting** is the inability of the model to predict well the labels of the data it was trained on.

What's the reasons cause underfitting?

- 1) your model is too simple for the data. ((for example, a linear model can often underfit))
- 2) the features you engineered are not informative enough.

  Solve 1 by more complex model, and 2 by engineer features with higher predictive power.

- **Overfitting** predicts very well the training data but poorly the data from at least one of the two holdout sets (validation and test)

What's the reasons cause overfitting?

- 1) your model is too complex for the data. ((for example, a very tall decision tree or a very deep or wide neural network often overfit))
- 2) you have too many features but a small number of training examples.

  Solutions a) Try a simpler model b) Reduce the dimensionality c) Add more training data d) Regularize the model.

- **Regularization** is the most widely used approach to prevent overfitting.
- Regularization forces the learning algorithm to build a less complex model.
- Regularization adding a penalizing term whose value is higher when the model is more complex.

Recall the linear regression objective:

$$\min_{\mathbf{w},b} \frac{1}{N}\sum_{i=1}^{N}(f_{\mathbf{w},b}(\mathbf{x}_i) - y_i)^2.$$

An L1-regularized objective looks like this:

lasso

$$\min_{\mathbf{w},b}\left[C|\mathbf{w}| + \frac{1}{N}\sum_{i=1}^{N}(f_{\mathbf{w},b}(\mathbf{x}_i) - y_i)^2\right],$$

- Your role as the data analyst is to find such a value of the hyperparameter C that doesn’t increase the bias too much but reduces the variance to a level reasonable for the problem at hand.

An L2-regularized objective looks like this:

Ridge

$$\min_{\mathbf{w},b}\left[C\|\mathbf{w}\|^2 + \frac{1}{N}\sum_{i=1}^{N}(f_{\mathbf{w},b}(\mathbf{x}_i) - y_i)^2\right], \quad \text{where } \|\mathbf{w}\|^2 \overset{\text{def}}{=} \sum_{j=1}^{D}(w^{(j)})^2. \qquad (4)$$

- L1 regularization produces a sparse model a model that has most of its parameters (in case of linear models, most of w (j)) equal to zero.
- L1 performs feature selection by deciding which features are essential for prediction and which are not. useful in case you want to increase model **explainability**.
- L2 usually gives better results. L2 also has the advantage of being differentiable, so gradient descent can be used for optimizing the objective function.
- L1 and L2 regularization methods were also combined in what is called **elastic net regularization.**

- MSE Test > MSE training => sign of overfitting

Determine metrics to assess the classification model?

- 1) confusion matrix
- 2) accuracy
- 3) cost-sensitive accuracy
- 4) precision/recall
- 5) area under the ROC curve

- Confusion matrix is used to calculate two other performance metrics: precision and recall.

	spam (predicted)	not_spam (predicted)
spam (actual)	23 (TP)	1 (FN)
not_spam (actual)	12 (FP)	556 (TN)

- **Precision** is the ratio of correct positive predictions to the overall number of positive predictions:

$$\text{precision} \stackrel{\text{def}}{=} \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

- So, Precision $= \frac{23}{23+12} = 0.65$
- **Recall** is the ratio of correct positive predictions to the overall number of positive examples in the dataset:

$$\text{recall} \stackrel{\text{def}}{=} \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

- So, recall $= \frac{23}{23+1} = 0.95$
- Prediction problem of research of documents in the database using a query. The precision is the proportion of relevant documents in the list of all returned documents. The recall is the ratio of the relevant documents returned by the search engine to the total number of the relevant documents that could have been returned.
- In the case of the spam detection problem. We want to have high precision and lower recall.
- Almost always, in practice, we have to choose between a high precision or a high recall.

- **Accuracy** is given by the number of correctly classified examples divided by the total number of classified examples.

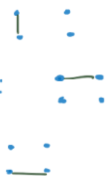
$$\text{accuracy} \overset{\text{def}}{=} \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}.$$

- So, Accuracy $= \frac{23+556}{23+556+12+1} = 0.97$
- Accuracy is a useful metric when errors in predicting all classes are equally important.

1) Precision :

2) recall = TPR :

3) FPR :



- The ROC curve (stands for "receiver operating characteristic;")
- ROC curves use a combination of the true positive rate TPR (defined exactly as recall) and false positive rate FPR.

$$\text{recall} \longleftarrow \text{TPR} \overset{\text{def}}{=} \frac{\text{TP}}{\text{TP} + \text{FN}} \text{ and } \text{FPR} \overset{\text{def}}{=} \frac{\text{FP}}{\text{FP} + \text{TN}}. \longleftarrow$$ الصف الاخير

- The higher the area under the ROC curve (AUC), the better the classifier. A classifier with an AUC higher than 0.5 is better than a random classifier. If AUC is lower than 0.5, then something is wrong with your model. A perfect classifier would have an AUC of 1
- if your model behaves well, you obtain a good classifier by selecting the value of the threshold that gives **TPR close to 1** while **keeping FPR near 0.**
- Hyperparameter such as such as and d for ID3, C for SVM, or α for gradient descent.
- For **Hyperparameter tuning** one typical way to do that **grid search** by trying all combinations of hyperparameters
- SVM hyperparametes C (a positive real number) and the kernel (either "linear" or "rbf")
- There are more efficient techniques, such as **random search** and **Bayesian hyperparameter optimization**.
- **Random search** you provide a statistical distribution for each hyperparameter from which values are randomly sampled and set the total number of combinations you want to try.

- **Bayesian techniques** use past evaluation results to choose the next values to evaluate. The idea is to limit the number of expensive optimizations.
- **Cross-validation** works as follows. First, you fix the values of the hyperparameters you want to evaluate. Then you split your training set into several subsets of the same size. Each subset is called a fold. Typically, five-fold cross-validation is used in practice.
- Each Fk, k = 1, . . . , 5 contains 20% of your training data.
- To train the first model, f1, you use all examples from folds F2, F3, F4, and F5 as the training set and the examples from F1 as the validation set.
- You continue building models iteratively like this and compute the value of the metric of interest on each validation set, from F1 to F5. Then you average the five values of the metric to get the final value
- You can use grid search with cross-validation to find the best values of hyperparameters.

- A neural network (NN), just like a regression or an SVM model, is a mathematical function: $y = f_{NN}(\mathbf{x})$.

- So, for a 3-layer neural network that returns a scalar, fNN looks like this: $y = f_{NN}(\mathbf{x}) = f_3(\boldsymbol{f}_2(\boldsymbol{f}_1(\mathbf{x})))$.

- The function g_l is called an activation function. It is a fixed, usually nonlinear function chosen by the data analyst before the learning is started.

- multilayer perceptron MLP and often referred to as a vanilla neural network.

- **activation function** last layer determines if it’s regression or a classification model,

- The neural network is represented graphically as a connected combination of **units** logically organized into one or more **layers**. Each unit is represented by either a circle or a rectangle.

- Usually, all units of a layer use the same activation function, but it’s not a rule.

- Deep learning refers to training neural networks with more than two non-output layers.

- **Backpropagation** is an efficient algorithm for computing gradients on neural networks using the chain rule.