Approach towards Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning

Mrs .Swati Dhabarde
Department of Computer Science & Engineering
Priyadarshini Indira Gandhi College of Enginering
Email:-swati.dhabarde@gmail.com

Abstract:-

Machine learning is the subfield of AI concerned with intelligent systems that learn. To understand machine learning, it is helpful to have a clear notion of intelligent systems .The correct use of model evaluation, model selection, and algorithm selection techniques is vital in academic machine learning research as well as in many industrial settings. This paper reviews different techniques that can be used for each of these three subtasks and discusses the main advantages and disadvantages of each technique with references to theoretical and empirical studies [1,2].

Common methods such as the holdout method for model evaluation and selection are discuss, which are not recommended when working with small datasets. Different flavors of the bootstrap technique are introduced for estimating the uncertainty of performance estimates, as an alternative to confidence intervals via normal approximation if bootstrapping is computationally feasible. Common cross-validation techniques such as leave-one- out cross-validation and k-fold cross-validation are reviewed, the bias-variance trade-off for choosing k is discussed, and practical tips for the optimal choice of k are given based on empirical evidence [1,2].

Keywords:-

Bias,Variance,hyperparameter,supervised ,unsupervised.

1 Introduction

The goal of modelling is to derive new information based on the data available, to describe the data by summarizing it in some appropriate way, or to make predictions about future data values. Models are used across the fields of science ranging from computer science to physics, and economics to engineering. The principal idea of model selection is to estimate the performance of different model candidates in order to choose the best model achievable (Hastie et al. 2009). The improved model performance achieved with model selection often brings along more reliable functioning of the model, better predictions about future outcomes of the process, financial savings
or increased safety in safety-critical applications.[1,2]

."How do we estimate the performance of a machine learning model?" the answer to this question might be as follows: "First, we feed the training data to our learning algorithm to learn a model. Second, we predict the labels of our test set. Third, we count the number of wrong predictions on the test dataset to compute the model's prediction accuracy." Depending on our goal, however, estimating the performance of a model is not that trivial, unfortunately. Maybe we should address the previous question from a different angle: "Why do we care about performance estimates at all?" Ideally, the estimated performance of a model tells how well it performs on unseen data – making predictions on future data is often

the main problem we want to solve in applications of machine learning or the development of new algorithms.[1][2]

Typically, machine learning involves a lot of experimentation, though – for example, the tuning of the internal knobs of a learning algorithm, the so-called hyperparameters. Running a learning algorithm over a training dataset with different hyperparameter settings will result in different models. Since we are typically interested in selecting the best-performing model from this set, we need to find a way to estimate their respective performances in order to rank them against each other. Going one step beyond mere algorithm fine-tuning, we are usually not only experimenting with the one single algorithm that we think would be the "best solution" under the given circumstances. More often than not, we want to compare different algorithms to each other, oftentimes in terms of predictive and computational performance. Let us summarize the main points why we evaluate the predictive performance of a model:

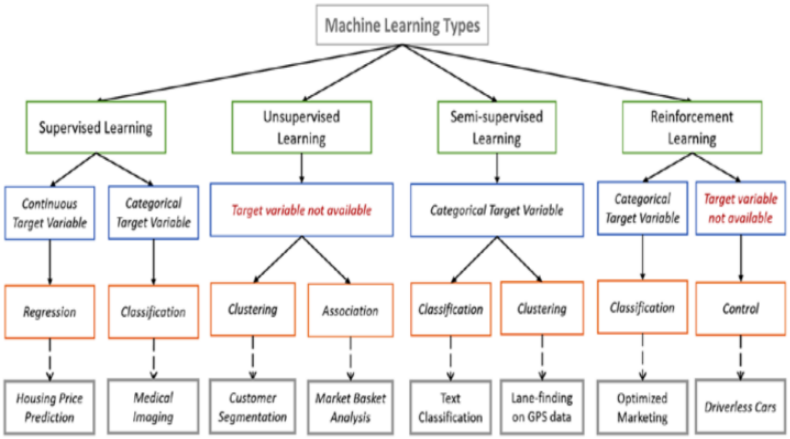
1. We want to estimate the generalization performance, the predictive performance of our model on future (unseen) data.
2. We want to increase the predictive performance by tweaking the learning algorithm and selecting the best performing model from a given hypothesis space.
3. We want to identify the machine learning algorithm that is best-suited for the problem at hand; thus, we want to compare different algorithms, selecting the best-performing one as well as the best performing model from the algorithm's hypothesis space.[1,2]

1.1Assumptions and Terminology

Model evaluation is certainly a complex topic. To make sure that we do not diverge too much from the core message, let us make certain assumptions and go over some of the technical terms that we will use throughout this paper.

1.1.1. Supervised learning and classification

Machine Learning is the field of study that gives computers the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one would have ever come across. As it is evident from the name, it gives the computer that which makes it more similar to humans: The ability to learn. Machine learning is actively being used today, perhaps in many more places than one would expect



In Supervised Learning, algorithms learn from labelled data. After understanding the data, the algorithm determines which label should be given to new data based on pattern and associating the patterns to the unlabeled new data[1].
Supervised Learning can be divided into 2 categories i.e Classification & Regression
Classification predicts the a category the data belongs to.eg: Spam Detection, Churn Prediction, Sentiment Analysis,Dog Breed Detection.Regression predicts a numerical value based on previous observed data.eg: House Price Prediction, Stock Price Prediction, Height-Weight Prediction.

0-1 loss and prediction accuracy

The lower the loss, the better a model (unless the model has over-fitted to the training data). The loss is calculated on training and validation and its interperation is how well the model is doing for these two sets. Unlike accuracy, loss is not a percentage. It is a summation of the errors made for each example in training or validation sets.

Bias.

Bias is the difference between the average prediction of our model and the correct value which we are trying to predict. Model with high bias pays very little attention to the training data and oversimplifies the model. It always leads to high error on training and test data.[4]
Low Bias: Suggests less assumptions about the form of the target function.
High-Bias: Suggests more assumptions about the form of the target function.
Examples of low-bias machine learning algorithms include: Decision Trees, k-Nearest Neighbors and Support Vector Machines.Examples of high-bias machine learning algorithms include: Linear Regression, Linear Discriminant Analysis and Logistic Regression.

Variance Error

Variance is the amount that the estimate of the target function will change if different training data was used. Variance is the variability of model prediction for a given data point or a value which tells us spread of our data. Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn't seen before. As a result, such models perform very well on training data but has high error rates on test data.[4]

The target function is estimated from the training data by a machine learning algorithm, so we should expect the algorithm to have some variance. Ideally, it should not change too much from one training dataset to the next, meaning that the algorithm is good at picking out the hidden underlying mapping between the inputs and the output variables.

Machine learning algorithms that have a high variance are strongly influenced by the specifics of the training data. This means that the specifics of the training have influences the number and types of parameters used to characterize the mapping function.

Low Variance: Suggests small changes to the estimate of the target function with changes to the training dataset.
High Variance: Suggests large changes to the estimate of the target function with changes to the training dataset.

Generally, nonparametric machine learning algorithms that have a lot of flexibility have a high variance. For example, decision trees have a high variance, that is even higher if the trees are not pruned before use.Examples of low-variance machine learning algorithms include: Linear Regression, Linear Discriminant Analysis and Logistic Regression.Examples

of high-variance machine learning algorithms include: Decision Trees, k-Nearest Neighbors and Support Vector Machines.

1.2Bias-Variance Trade-Off

The goal of any supervised machine learning algorithm is to achieve low bias and low variance. In turn the algorithm should achieve good prediction performance.
You can see a general trend in the examples above:
Parametric or linear machine learning algorithms often have a high bias but a low variance.
Non-parametric or non-linear machine learning algorithms often have a low bias but a high variance.The parameterization of machine learning algorithms is often a battle to balance out bias and variance.[1,4]

1.3 Mathematically:-

Let the variable we are trying to predict as Y and other covariates as X. We assume there is a relationship between the two such that
Y=f(X) + eWhere e is the error term and it's normally distributed with a mean of 0.We will make a model f^(X) of f(X) using linear regression or any other modeling technique.So the expected squared error at a point x is
So the expected squared error at a point x is

$$Err(x) = E\left[(Y - \hat{f}\,(x))^2\right]$$

The Err(x) can be further decomposed as

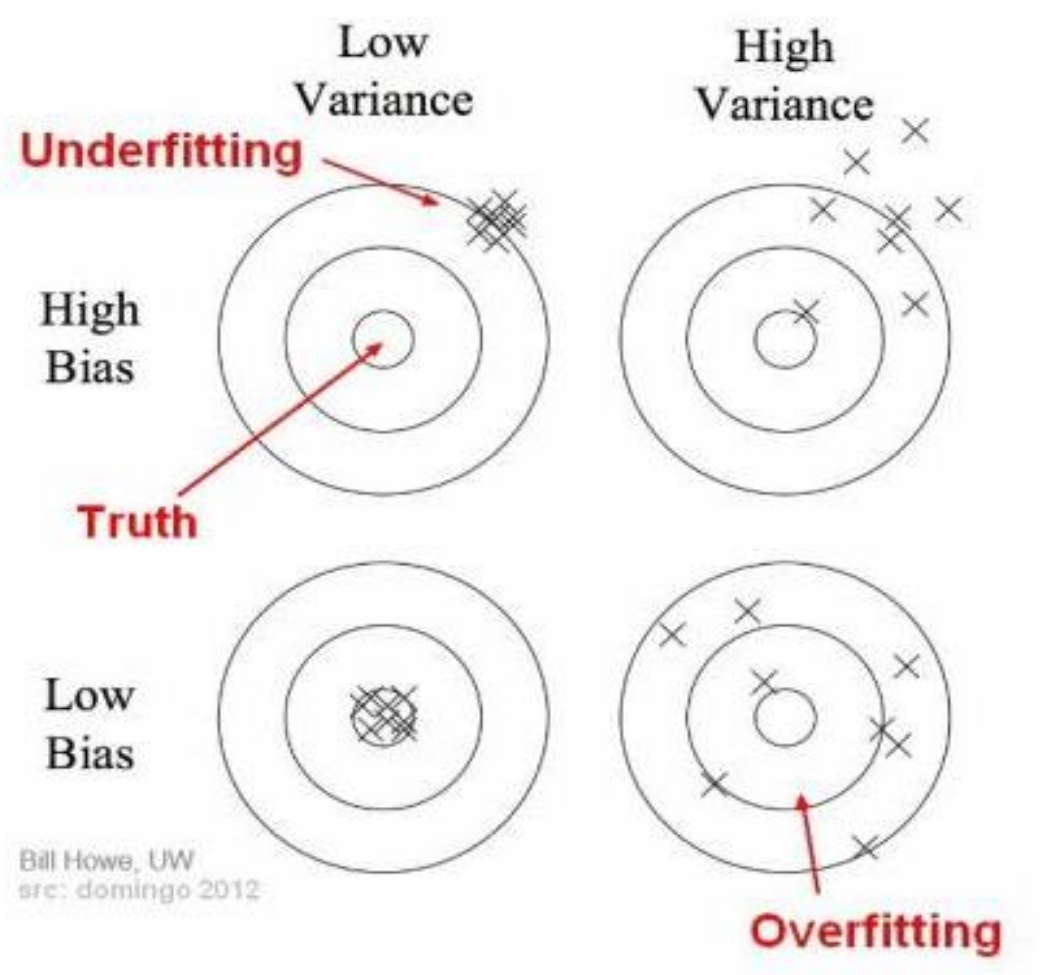
$$Err(x) = \left(E[\hat{f}\,(x)] - f(x)\right)^2 + E\left[\left(\hat{f}\,(x) - E[\hat{f}\,(x)]\right)^2\right] + \sigma_e^2$$

$$Err(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

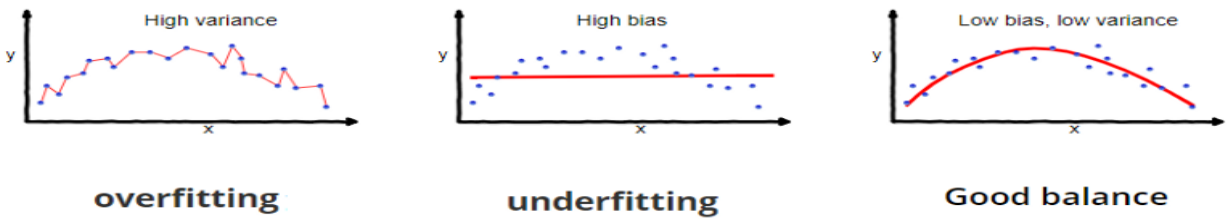
Err(x) is the sum of Bias², variance and the irreducible error.Irreducible error is the error that can't be reduced by creating good models. It is a measure of the amount of noise in our data. Here it is important to understand that no matter how good we make our model, our data will have certain amount of noise or irreducible error that can not be removed.[1,4]

1.4Bias and variance using bulls-eye diagram:-

In the below diagram, center of the target is a model that perfectly predicts correct values. As we move away from the bulls-eye our predictions become get worse and worse. We can repeat our process of model building to get separate hits on the target.In supervised learning, underfitting happens when a model unable to capture the underlying pattern of the data. These models usually have high bias and low variance. It happens when we have very less amount of data to build an accurate model or when we try to build a linear model with a nonlinear data. Also, these kind of models are very simple to capture the complex patterns in data like Linear and logistic regression[8,9].



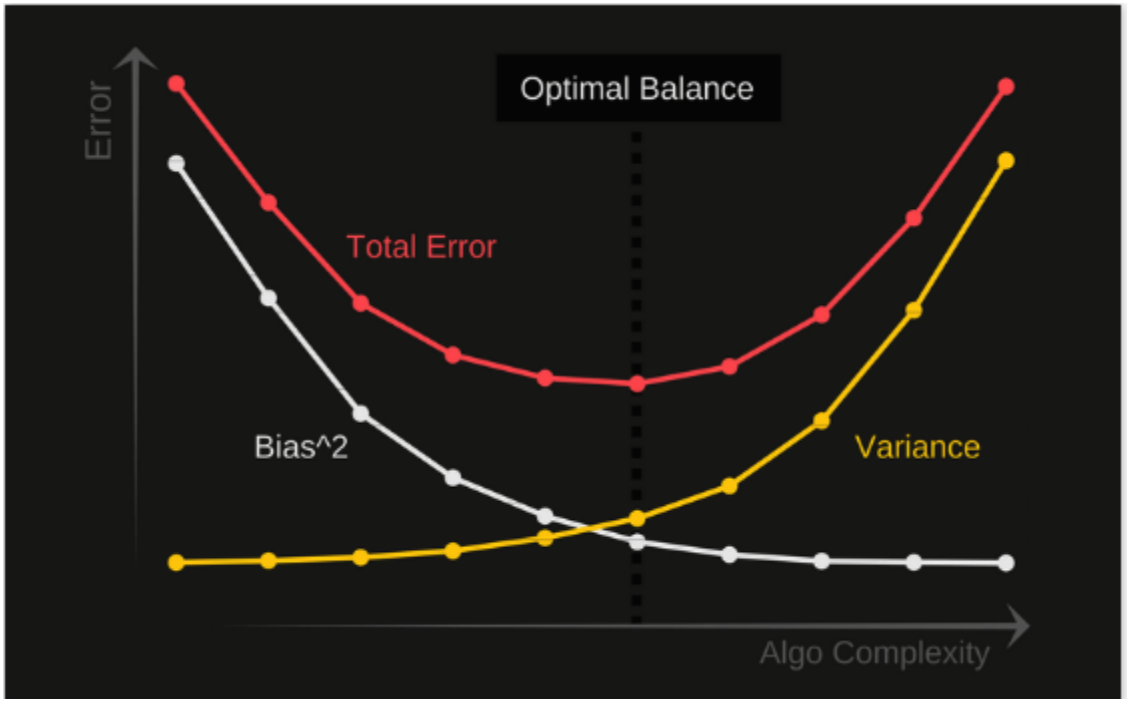
In supervised learning, overfitting happens when our model captures the noise along with the underlying pattern in data. It happens when we train our model a lot over noisy dataset. These models have low bias and high variance. These models are very complex like Decision trees which are prone to overfitting. [7]



1.5Why is Bias Variance Tradeoff:

If our model is too simple and has very few parameters then it may have high bias and low variance. On the other hand if our model has large number of parameters then it's going to have high variance and low bias. So we need to find the right/good balance without overfitting and underfitting the data.[4][1]This tradeoff complex at the same time.

Total Error To build a good model, we need to find a good balance between bias and variance such that it minimizes the total error.



An optimal balance of bias and variance would never overfit or underfit the model.

Therefore understanding bias and variance is critical for understanding the behavior of prediction models.

Below are two examples of configuring the bias-variance trade-off for specific algorithms:
The k-nearest neighbors algorithm has low bias and high variance, but the trade-off can be changed by increasing the value of k which increases the number of neighbors that contribute t the prediction and in turn increases the bias of the model.
The support vector machine algorithm has low bias and high variance, but the trade-off can be changed by increasing the C parameter that influences the number of violations of the margin allowed in the training data which increases the bias but decreases the variance.
There is no escaping the relationship between bias and variance in machine

learning.Increasing the bias will decrease the variance.Increasing the variance will decrease the bias.

There is a trade-off at play between these two concerns and the algorithms you choose and the way you choose to configure them are finding different balances in this trade-off for your problem

In reality, we cannot calculate the real bias and variance error terms because we do not know the actual underlying target function.

Learning algorithm. Again, our goal is to find or approximate the target function, and the learning algorithm is a set of instructions that tried to model the target function using a training dataset. A learning algorithm comes with a hypothesis space, the set of possible hypotheses it can explore to model the unknown target function by formulating the final hypothesis.Hyperparameters. Hyperparameters are the tuning parameters of a machine learning algorithm –

for example, the regularization strength of an L2 penalty in the loss function of logistic regression, or a value for setting the maximum depth of a decision tree classifier. In contrast, model parameters are the parameters that a learning algorithm fits to the training data – the parameters of the model itself. For example, the weight coefficients (or slope) of a linear regression line and its bias term (here: y-axis intercept) are model parameters.

2. Resubstitution Validation and the Holdout Method

The holdout method is inarguably the simplest model evaluation technique; it can be summarized as follows. First, we take a labeled dataset and split it into two parts: A training and a test set. Then, we fit a model to the training data and predict the labels of the test set. The fraction of correct predictions, which can be computed by comparing the predicted labels to the ground truth labels of the test set, constitutes our estimate of the model's prediction accuracy. Here, it is important to note that we do not want to train and evaluate a model on the same training dataset (this is called resubstitution validation or resubstitution evaluation), since it would typically introduce a very optimistic bias due to overfitting. In other words, we cannot tell whether the model simply memorized the training data, or whether it generalizes well to new, unseen data. (On a side note, we can estimate this so-called optimism bias as the difference between the training and test accuracy.)

Typically, the splitting of a dataset into training and test sets is a simple process of random subsam- pling. We assume that all data points have been drawn from the same probability distribution (with respect to each class). And we randomly choose 2/3 of these samples for the training set and 1/3 of the samples for the test set.

3. Bootstrapping and Uncertainties

The previous section (Section 1, Introduction: Essential Model Evaluation Terms and Techniques) introduced the general ideas behind model evaluation in supervised machine learning. We discussedthe holdout method, which helps us to deal with real world limitations such as limited access to new,labeled data for model evaluation. Using the holdout method, we split our dataset into two parts: Atraining and a test set. First, we provide the training data to a supervised learning algorithm. The learning algorithm builds a model from the training set of labeled observations. Then, we evaluate thepredictive performance of the model on an independent test set that shall represent new, unseen data.Also, we briefly introduced the normal approximation, which requires us to make certain assumptions that allow us to compute confidence intervals for modeling the uncertainty of our performance estimate based on a single test set, which we have to take with a grain of salt.This section introduces some of the advanced techniques for model evaluation. We will start bydiscussing techniques for estimating the uncertainty of our estimated model performance as well as the model's

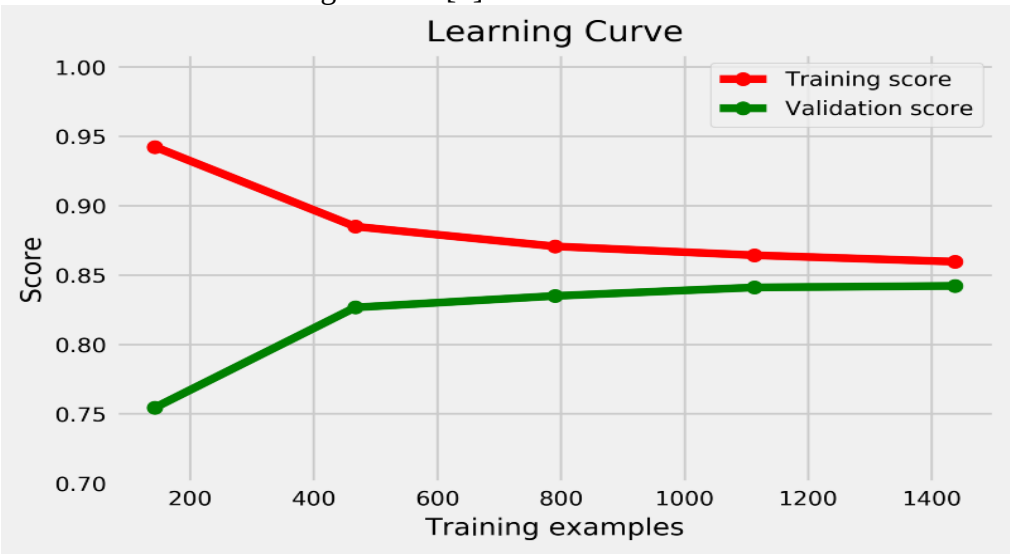
variance and stability. And after getting these basics under our belt, we will look atcross-validation techniques for model selection in the next article in this series. As we remember fromSection 1, there are three related, yet different tasks or reasons why we care about model evaluation:

1. We want to estimate the generalization accuracy, the predictive performance of a model on future (unseen) data.
2. We want to increase the predictive performance by tweaking the learning algorithm and selecting the best-performing model from a given hypothesis space.
3. We want to identify the machine learning algorithm that is best-suited for the problem at hand. Hence, we want to compare different algorithms, selecting the best-performing one aswell as the best-performing model from the algorithm's hypothesis.

4. Learning curves

Learning curves, and why they are useful But why should we retrain the model after model selection / model evaluation? The answer is best illustrated using learning curves. In a learning curve, the performance of a model both on the training and validation set is plotted as a function of the training set size. Fig. 1 shows a typical learning curve: The training score (performance on the training set) decreases with increasing training set size while the validation score increases at the same time. High training score and low validation score at the same time indicates that the model has overfit the data, i.e., has adapted too well to the specific training set samples. As the training set increases, overfitting decreases, and the validation score increases.

Especially for data-hungry machine learning models, the learning curve might not yet have reached a plateau at the given training set size, which means the generalization error might still decrease when providing more data to the model. Hence, it seems reasonable to increase the training set (by adding the validation set) before estimating the generalization error on the test set, and to further take advantage of the test set data for model fitting before shipping the model. Whether or not this strategy is needed depends strongly on the slope of the learning curve at the initial training set size.[3]



Learning curves further allow to easily illustrate the concept of (statistical) *bias* and *variance*. Bias in this context refers to erroneous (e.g. simplifying) model assumptions, which can cause the model to underfit the data. A high-bias model does not adequately capture the structure present in the data. Variance on the other hand quantifies how much the model varies as we change the training data. A high-variance model is very sensitive to small fluctuations in the training data, which can cause the model to overfit. The amount of bias

and variance can be estimated using learning curves: A model exhibits high variance, but low bias if the training score plateaus at a high level while the validation score at a low level, i.e., if there is a large gap between training and validation score. A model with low variance but high bias, in contrast, is a model where both training and validation score are low, but similar. Very simple models are high-bias, low-variance while with increasing model complexity they become low-bias, high-variance.

Have your cake and eat it too: cross-validation for hyperparameter selection

For hyperparameter selection, we can use K-fold cross-validation (CV). Cross-validation works as follows:

We split the training set into K smaller sets. Note that the caveats regarding imbalanced data also apply here.We set aside each of the K folds one time, as illustrated in Fig. 2. We train as many models as there are different combinations of model hyperparameters on the remaining K-1 folds and compute the validation score on the hold-out fold.

For each set of hyperparameters we compute the mean validation score and select the hyperparameter set with best performance on the hold-out validation sets. Alternatively, we can apply the "one-standard-error-rule" [2], which means that we choose the most parsimonious model (the model with lowest complexity) whose performance is not more than a standard error below the best performing model.[3]

Subsequently, we train the model with the chosen hyperparameter set on the full training set and estimate the generalization error using the test set. Lastly, we retrain the model using the combined data of training and test set.

Fig. 2: Illustration of 5-fold cross-validation.

How many splits should we make, i.e., how should we choose K? Unfortunately, there is no free lunch, i.e., no single answer that always works best. If we choose K=N where N is the number of training examples, we are dealing with a method called leave-one-out cross-validation (LOOCV). The advantage here is that since we always use almost the entire data for training, the estimated prediction performance is approximately unbiased, meaning that the difference between expected value of the prediction error and "true" prediction error is very low. The disadvantage, however, is that LOOCV is computationally expensive and the variance can be high, meaning that our prediction performance estimate can fluctuate strongly around its "true" value. In contrast, if we choose K=5 or K=10, the variance of our

prediction performance estimate is low, but our estimate might be overly pessimistic since we use only 80–90% of the available data for training (cf. discussion of the learning curve above). Nevertheless, 10-fold (or 5-fold) CV is recommended as a rule of thumb [2].Inside the matryoshka: nested cross-validation for algorithm selection

For algorithm selection we need a more elaborate method. Here, nested cross-validation comes to the rescue, which is illustrated in Fig. 3 and works as follows:

We split the data into K smaller sets (outer fold).Each of the K folds we set aside one time. For each learning method we then perform K'-fold CV (following the procedure above) on the K-1remaining folds, in which we do we do hyperparameter selection. For brevity, one denotes nested CV with K outer folds and K' inner folds as KxK'nested CV. Typical values for KxK' are 5x2 or 5x3.We use the best hyperparameter set for each algorithm to estimate its validationscore on the hold-out fold.Then we compute the mean validation score (as well as standard deviation) over the K folds and select the best performing algorithmSubsequently, we choose the best hyperparameter set based on CV using the full training set and estimate the generalization error using the test set.Lastly, we retrain the model using the combined data of training and test set[5].

5. Confusion matrix

A confusion matrix is a table that is often used to **describe the performance of a classification model** (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.

I wanted to create a **"quick reference guide" for confusion matrix terminology** because I couldn't find an existing resource that suited my requirements: compact in presentation, using numbers instead of arbitrary variables, and explained both in terms of formulas and sentences.[6]

Let's start with an **example confusion matrix for a binary classifier** (though it can easily be extended to the case of more than two classes):

n=165	Predicted: NO	Predicted: YES
Actual: NO	50	10
Actual: YES	5	100

What can we learn from this matrix?

- There are two possible predicted classes: "yes" and "no". If we were predicting the presence of a disease, for example, "yes" would mean they have the disease, and "no" would mean they don't have the disease.
- The classifier made a total of 165 predictions (e.g., 165 patients were being tested for the presence of that disease).
- Out of those 165 cases, the classifier predicted "yes" 110 times, and "no" 55 times.
- In reality, 105 patients in the sample have the disease, and 60 patients do not.[6]

Let's now define the most basic terms, which are whole numbers (not rates):

- **true positives (TP):** These are cases in which we predicted yes (they have the disease), and they do have the disease.
- **true negatives (TN):** We predicted no, and they don't have the disease.
- **false positives (FP):** We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")
- **false negatives (FN):** We predicted no, but they actually do have the disease. (Also known as a "Type II error.")

I've added these terms to the confusion matrix, and also added the row and column totals:

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

This is a list of rates that are often computed from a confusion matrix for a binary classifier:

- **Accuracy:** Overall, how often is the classifier correct?
  - (TP+TN)/total = (100+50)/165 = 0.91
- **Misclassification Rate:** Overall, how often is it wrong?
  - (FP+FN)/total = (10+5)/165 = 0.09
  - equivalent to 1 minus Accuracy
  - also known as "Error Rate"
- **True Positive Rate:** When it's actually yes, how often does it predict yes?
  - TP/actual yes = 100/105 = 0.95
  - also known as "Sensitivity" or "Recall"
- **False Positive Rate:** When it's actually no, how often does it predict yes?
  - FP/actual no = 10/60 = 0.17
- **True Negative Rate:** When it's actually no, how often does it predict no?
  - TN/actual no = 50/60 = 0.83
  - equivalent to 1 minus False Positive Rate
  - also known as "Specificity"
- **Precision:** When it predicts yes, how often is it correct?
  - TP/predicted yes = 100/110 = 0.91
- **Prevalence:** How often does the yes condition actually occur in our sample?
  - actual yes/total = 105/165 = 0.64

6. Application of Machine Learning

- Virtual Personal Assistants
- Predictions while Commuting
- Videos Surveillance
- Social Media Services
- Email Spam and Malware Filtering
- Online Customer Support
- Search Engine Result Refining
- Product Recommendations
- Online Fraud Detection

7. Future Aspects

In the post-industrialization era, people have worked to create a machine that behaves like a human. The thinking machine is AI's biggest gift to humankind; the grand entry of this self-propelled machine has suddenly changed the operative rules of business. In the recent years, self-driving vehicles, digital assistants, robotic factory staff, and smart cities have proven that intelligent machines are possible. AI has transformed most industry sectors like retail, manufacturing, finance, healthcare, and media and continues to invade new

territories.ML will be an integral part of all AI systems, large or small. All these metrics were important to calculate the performance of every problem statement efficiently.

8. Conclusion:-

In this paper we discuss various methods for bias variance tradeoff. There are many ways for evaluating the generalization performance of predictive models. So far,this paper covered the holdout method, different flavors of the bootstrap approach, Confusion matrix,ROC curves and k-fold crossvalidation.Using holdout method is absolutely fine for model evaluation when working with relatively large sample sizes.

References:-

[1] Model evaluation, model selection, and algorithm selection in machine learning by Sebastian Raschka.
[2] Hastie T., Tibshirani R., and Friedman J., The Elements of Statistical Learning, New York, NY, USA: Springer New York Inc. (2008).
[3] https://towardsdatascience.com/a-short-introduction-to-model-selection-bb1bb9c73376.
[4] https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229.
[5] https://heartbeat.fritz.ai/model-evaluation-selection-i-30d803a44ee
[6] https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/
[7] https://medium.com/fintechexplained/the-problem-of-overfitting-and-how-to-resolve-it-1eb9456b1dfd
[8] Kohavi, R., 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In Ijcai (Vol. 14, No. 2, pp. 1137-1145). [Citation source] [PDF]
[9] Efron, B. and Tibshirani, R.J., 1994. An introduction to the bootstrap. CRC press. [Citation source] [PDF]
[10] [1] W. Richert, L. P. Coelho, "Building Machine Learning Systems withPython", Packt Publishing Ltd., ISBN 978-1-78216-140-0
[11] M. Welling, "A First Encounter with Machine Learning"
[12] M. Bowles, "Machine Learning in Python: Essential Techniques forPredictive Analytics", John Wiley & Sons Inc., ISBN: 978-1-118-96174-2
[13] S.B. Kotsiantis, "Supervised Machine Learning: A Review ofClassification Techniques", Informatica 31 (2007) 249-268
[14] L. Rokach, O. Maimon, "Top – Down Induction of Decision TreesClassifiers – A Survey", IEEE Transactions on Systems,
[15] D. Lowd, P. Domingos, "Naïve Bayes Models for ProbabilityEstimation"
[16] https://webdocs.cs.ualberta.ca/~greiner/C-651/Homework2_Fall2008.html.
[17] D. Meyer, "Support Vector Machines – The Interface to libsvm inpackage e1071", August 2015
[18] S. S. Shwartz, Y. Singer, N. Srebro, "Pegasos: Primal Estimated sub -Gradient Solver for SVM", Proceedings of the 24th InternationalConference on Machine Learning, Corvallis, OR, 2007
[19] http://www.simplilearn.com/what-is-machine-learning-and-why-itmatters-article
[20] P. Harrington, "Machine Learning in action", Manning PublicationsCo., Shelter Island, New York, 2012.
[21] https://frnsys.com/ai_notes/machine_learning/model_selection.html