MACHINE LEARNING

ASSIGNMENT - 5

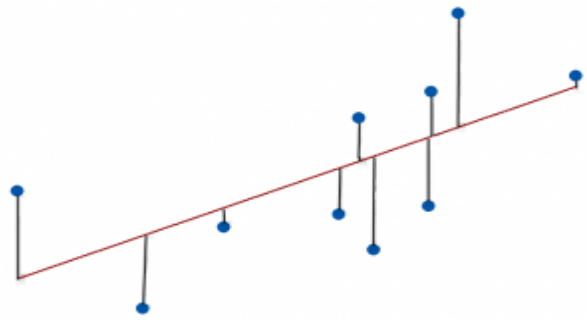
Q1 to Q15 are subjective answer type questions, Answer them briefly.

---

1. R-squared or Residual Sum of Squares (RSS) which one of these two is a better measure of goodness of fit model in regression and why?

Ans: R-squared is a goodness of fit model for linear regression, which indicates the percentage of variance in the dependent variable that the independent variable explains collectively. R-squared measures the strength of the relationship between our model and the dependent variable on the scale of 0-100%.
After we fit a linear regression model , we determine how well the model fits the data. For instance small R-squared values are always a problem and high R-squared values are usually not good.

Linear regression identifies the equation that produces the smallest difference between all the observed values and their fitted values.



Residuals are the distance between observed value and fitted value

Also, a regression model fits the data well if the differences between the observations and the predicted values are small and unbiased (fitted values are not too high or too low in observation space)
However, before assessing numeric values for goodness of fit (like R-squared), we need to evaluate the residual plots (for biased models).

R-squared evaluates the scatter of the data points around the fitted regression line. It is also called the **coefficient of determination,** or the coefficient of multiple determination for multiple regression. For the same data set, higher R-squared values represent smaller differences between the observed data and the fitted values.

R-squared is the percentage of the dependent variable variation that a linear model explains.

$$R^2 = \frac{\text{Variance explained by the model}}{\text{Total variance}}$$

R-squared is always between 0 and 100%:

- 0% represents a model that does not explain any of the variation in the response variable around its mean. The mean of the dependent variable predicts the dependent variable as well as the regression model.
- 100% represents a model that explains all the variation in the response variable around its mean.

Usually, the larger the R^2, the better the regression model fits our observations

R-squared, also known as the coefficient of determination, measures the proportion of variance in the dependent variable that is explained by the independent variables in the model. It ranges from 0 to 1, with a higher value indicating a better fit.

R-squared is useful for comparing different models or for determining the proportion of the variability in the dependent variable that is explained by the model.

Residual Sum of Squares (RSS) measures the total amount of unexplained variance in the dependent variable that remains after the model has been fit. It is the sum of the squared differences between the actual and predicted values of the dependent variable. A lower value of RSS indicates a better fit. RSS is useful for evaluating the accuracy of the predictions of the model.

Conclusion:
In general, both R-squared and RSS are important and are considered together when evaluating the goodness of fit of a model. However**, R-squared is often considered to be a better measure of goodness of fit than RSS** because it provides a single number that summarizes the proportion of variance in the dependent variable that is explained by the model, which is more interpretable and easier to compare across models.

---

2. What are TSS (Total Sum of Squares), ESS (Explained Sum of Squares) and RSS (Residual Sum of Squares) in regression. Also mention the equation relating these three metrics with each other.

Ans: TSS (Total sum of Squares)
TSS works as a cost function for a model which does not have an independent variable, but only y intercept (mean $\bar{y}$). This shows how good the model is without any independent variable. When independent variable is added the model performance is given by RSS.
The ration of RSS/TSS gives how good the model is as compared to the mean value without variance. Lesser the ratio,lesser is the residual error with actual values, and greater is the residual error with the mean. This implies that the model is more robust.
So, 1-RSS/TSS is considered as the measure of robustness of the model and is known as R^2

ESS(Explained Sum of Squares)

Explained sum of square (ESS) or **Regression sum of squares** is a statistical quantity used in modeling of a process. ESS gives an estimate of how well a model explains the observed data for the process.

It tells how much of the variation between observed data and predicted data is being explained by the model proposed. Mathematically, it is the sum of the squares of the difference between the predicted data and mean data.

Let $y_i = a + b_1x_{1i} + b_2x_{2i} + ... + \varepsilon_i$ is regression model, where:

y_i is the i^{th} observation of the response variable

x_{ji} is the i^{th} observation of the j^{th} explanatory variable

a and b_i are coefficients

i indexes the observations from 1 to n

ε_i is the i^{th} value of the error term

Then

$$\mathrm{ESS} = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2 .$$

ESS = total sum of squares – residual sum of squares

In general, higher the ESS value signifies greater amount of variation being explained by the model, which signifies a better model.

RSS(Residual Sum of Squares): The residual sum of squares (RSS) is a statistical technique used to measure the amount of variance in a data set that is not explained by a regression model. Instead, it estimates the variance in the residuals, or error term.

- The residual sum of squares (RSS) measures the level of variance in the error term, or residuals, of a regression model.
- The smaller the residual sum of squares, the better our model fits our data; the greater the residual sum of squares, the poorer our model fits our data.
- A value of zero means our model is a perfect fit

---

3. What is the need of regularization in machine learning?

Ans:Regularization is one of the most important concepts of machine learning. It is a technique to prevent the model from overfitting by adding some extra information to it.

Sometimes the machine learning model performs well with the training data ,and sometimes does not perform well with the test data. It means the model is not able to predict the output and does not give the required output and hence the model is called overfitted. This problem can be solved with regularization technique.

This technique can be used in such a way that it will allow to maintain all variables or features in the model by reducing the magnitude of the variables. Hence, it maintains accuracy as well as a generalization of the model.

It mainly regularizes or reduces the coefficient of features toward zero. In simple words, "*In regularization technique, we reduce the magnitude of the features by keeping the same number of features."*

There are mainly two regularization techniques:

- **Ridge Regression**
- **Lasso Regression**

Conclusion:

Regularization in Machine Learning is used to minimize the problem of overfitting, the result is that the model performs well on the unseen data(test data), once overfitting is minimized. Regularization will try to minimize a loss function by taking a penalty as shown below:

The residual sum of squares is our optimization function or loss function in simple linear regression (RSS).

$$\mathrm{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 .$$

Here ,

y is the dependent variable,
x_1, x_2, x_3,..... x_n are independent variables.
b_0, b_1 ,b_2..... b_n, are the coefficients estimates for different variables of x, these can also be called weights or magnitudes
Regularization will shrink these coefficients towards Zero,
Minimizing the loss means less error and model will be a good fit.

---

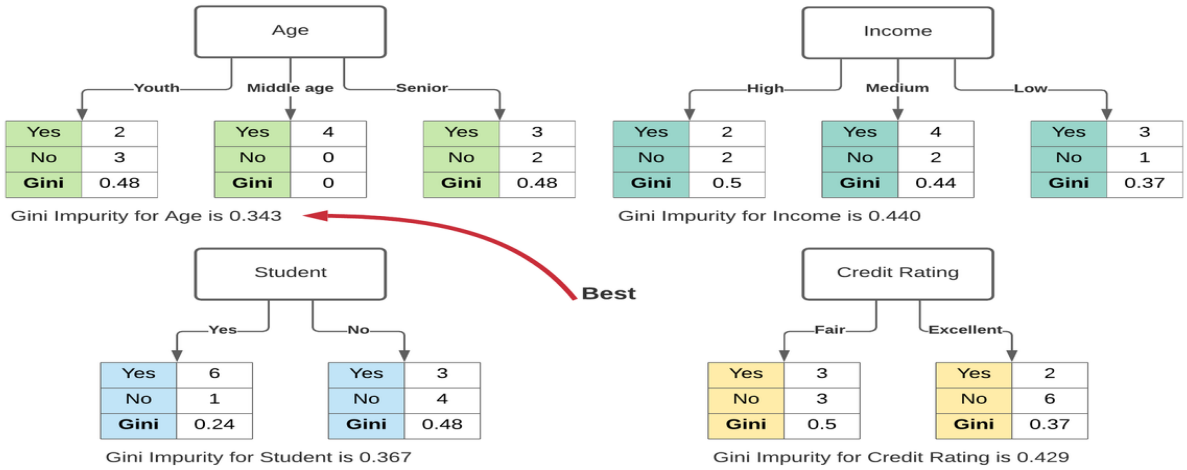
4. What is Gini–impurity index?

Ans: It is a measurement used to build Decision Trees to determine how the features of a dataset should split nodes to form the tree.

Also, the Gini Impurity of a dataset is a number between 0-0.5, which indicates the likelihood of new, random data being misclassified if it were given a random class label according to the class distribution in the dataset.

- The Gini Index is the additional approach to dividing a decision tree.
- Purity and impurity in a junction are the primary focus of the Entropy and Information Gain framework.
- The Gini Index, also known as Impurity, calculates the likelihood that somehow a randomly picked instance would be erroneously catalogued.

Example:



For example, if we want to build a classifier that determines if someone will default on their credit card. We have some labelled data with features, such as bins for age, income, credit rating, and whether or not each person is a student. To find the best feature for the first split of the tree – the root node – we can calculate how poorly each feature divided the data into the correct class, default ("yes") or didn't default ("no"). This calculation would measure the **impurity** of the split, and the feature with the lowest impurity would determine the best feature for splitting the current node.

This process would continue for each subsequent node using the remaining features.

In the image above, **age** has minumum gini impurity, so **age** is selected as the root in the decision tree.

Consider a dataset D that contains samples from k classes. The probability of samples belonging to class i at a given node can be denoted as pi. Then the Gini Impurity of D is defined as:

$$Gini(D) = 1 - \sum_{i=1}^{k} p_i^2$$

The node with uniform class distribution has the highest impurity. The minimum impurity is obtained when all records belong to the same class. Several examples are given in the following table to demonstrate the Gini Impurity computation.

---

5. Are unregularized decision-trees prone to overfitting? If yes, why?

Ans: Decision trees are prone to overfitting, especially when a tree is particularly deep. This is due to the amount of specificity we look at leading to smaller sample of events that meet the previous assumptions.

Overfitting happens when any learning processing overly optimizes training set error at the cost test error. While it's possible for training and testing to perform equality well in cross validation, it could be as the result of the data being very close in characteristics, which may not be a huge problem.

In the case of decision tree's, they can learn a training set to a point of high granularity that makes them easily overfit. Allowing a decision tree to split to a granular degree, is the behaviour of this model that makes it prone to learning every point extremely well — to the point of perfect classification — i.e.: overfitting.

---

6. What is an ensemble technique in machine learning?

Ans: In simple words, ensemble refers to a group of items. For e.g: a group of ministers, a group of dancers etc. An ensemble method is a technique which uses multiple independent similar or different models/weak learners to derive an output or make some predictions.

For e.g.

A random forest is an ensemble of multiple decision trees.

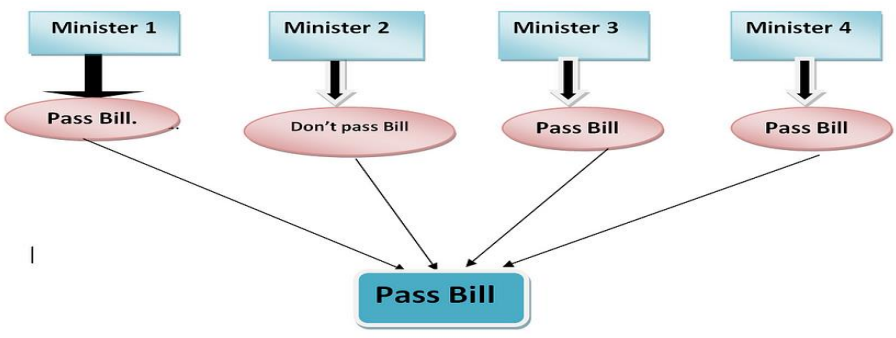
An ensemble can also be built with a combination of different models like random forest, SVM, Logistic regression etc**.**

Now, let's compare it with **a real-life example.**

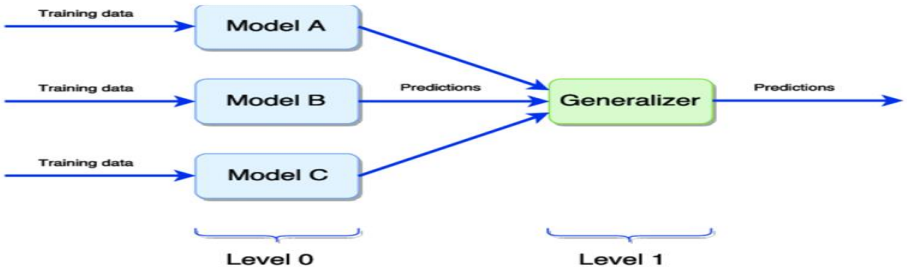
Suppose a bill is passed in the parliament and a meeting is held for the same with a group of ministers. Instead of the President or the prime minister taking the decision alone by themselves might not be a very good idea as this accounts for dictatorship and the results might not be so much in favour of the public. So, this can be referred to a single model like Decision Tree, Logistic Regression etc in machine learning and they will always slightly perform less better than the ensemble methods.

However, if a meeting is held between different ministers, let's say four of them, each of them will provide an opinion with pros and cons. And finally, the best opinion will be picked based on majority voting. This is how the ensemble technique works.

Minister 1: based on his pros and cons, says the bill should be passed.
Minister 2: based on his pros and cons, says the bill is not useful and might trigger a lot of challenges. Hence should not be passed.
Minister 3: Says bill can be passed.
Minister 4: Also says bill can be passed.



Similarly, we can see below that different models are used and based on the output of each, the final decision will be made on the majority voting.



As we know that, the errors and predictions in any machine learning models are adversely influenced by bias, variance and noise. In order to combat these disadvantages, ensemble methods are used.

Types of ensemble methods are:

1.Bagging
2.Boosting

---

7. What is the difference between Bagging and Boosting techniques?

Ans:

S.No	Bagging	Boosting
1	The simplest way of combining predictions that belong to the same type.	A way of combining predictions that belong to the different types.
2	Aim to decrease variance, not bias and solves overfitting issues in a model	Aim to decrease bias, not variance.
3	Each model receives equal weight.	Models are weighted according to their performance.
4	Each model is built independently.	New models are influenced by the performance of previously built models.
5	Different training data subsets are selected using row sampling with replacement and random sampling methods from the entire training dataset.	Every new subset contains the elements that were misclassified by previous models.
6	Bagging is usually applied where the classifier is unstable and has a high variance	Boosting is usually applied where the classifier is stable and simple and has high bias.

---

8. What is out-of-bag error in random forests?

Ans: OOB (out-of-bag) errors are an estimate of the performance of a random forest classifier or regressor on unseen data. In scikit-learn, the OOB error can be obtained using the oob_score_ attribute of the random forest classifier or regressor.

The OOB error is computed using the samples that were not included in the training of the individual trees. This is different from the error computed using the usual training and validation sets, which are used to tune the hyperparameters of the random forest.

The OOB error can be useful for evaluating the performance of the random forest on unseen data. It is not always a reliable estimate of the generalization error of the model, but it can provide a useful indication of how well the model is being performed.

---

9. What is K-fold cross-validation?

Ans: Cross-validation is a technique for evaluating ML models by training several ML models on subsets of the available input data and evaluating them on the complementary subset of the data. We use cross-validation to detect overfitting, I,e, failing to generalize a pattern.

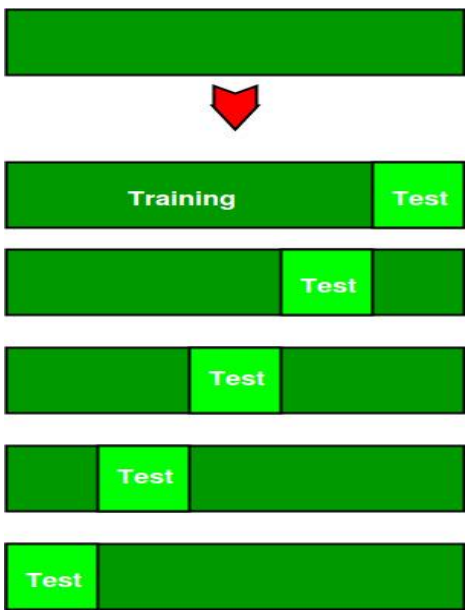
K-fold Cross-Validation is when the dataset is split into a **K** number of folds and is used to evaluate the model's ability when given new data. **K** refers to the number of groups the data sample is split into. For example, if we

see that the k-value is 5, we can call this a 5-fold cross-validation. Each fold is used as a testing set at one point in the process.

K-fold cross-validation steps:

1. Choose the k-value
2. Split the dataset into the number of k folds.
3. Start the process by using k-1 fold as the **test** dataset and the remaining folds as the **training** dataset
4. Train the model on the training dataset and validate it on the test dataset
5. Save the validation score
6. Repeat steps 3 – 5, but changing the value of the k test dataset. So we chose k-1 as our test dataset for the first round, we then move onto k-2 as the test dataset for the next rounds.
7. By the end of the process we have validated the model on every fold that we have.
8. Now ,average the results that were produced in step 5 to summarize the skill of the model.
9. We implement this by using the library ***sklearn.model_selection.KFold***

Example :The diagram below shows an example of the training subsets and evaluation subsets generated in k-fold cross-validation. Here, we have total 25 instances. In first iteration we use the first 20 percent of data for evaluation, and the remaining 80 percent for training([1-5] testing and [5-25] training) while in the second iteration we use the second subset of 20 percent for evaluation, and the remaining three subsets of the data for training([5-10] testing and [1-5 and 10-25] training), and so on.



Advantages of using K-fold cross-validation

- ✓ Overcome Overfitting: Cross validation helps to prevent overfitting by providing a good estimate of the model's performance on unseen data.
- ✓ Model Selection: Cross validation can be used to compare different models and select the one that performs the best on an average.

- ✓ Hyperparameter tuning: Cross validation can be used to optimize the hyperparameters of a model, such as the regularization parameter, by selecting the values that result in the best performance on the validation set.
- ✓ Data Efficient: Cross validation allows the use of all the available data for both training and validation, making it a more data-efficient method compared to traditional validation techniques.

---

10. What is hyper parameter tuning in machine learning and why it is done?

Ans: One traditional and popular way to perform hyperparameter tuning is by using an Exhaustive Grid Search from Scikit learn. This method tries every possible combination of each set of hyper-parameters. Using this method, we can find the best set of values in the parameter search space. This usually uses more computational power and takes a long time to run since this method needs to try every combination in the grid size.

---

11. What issues can occur if we have a large learning rate in Gradient Descent?

Ans: Gradient Descent works well when we set the learning rate to an appropriate value. This parameter **determines how fast or slow we will move towards the optimal weights**. If the learning rate is very large, we will skip the optimal solution. If it is too small, we will need too many iterations to converge to the best values. So, using a good learning rate is an advantage**.**

---

12. Can we use Logistic Regression for classification of Non-Linear Data? If not, why?

Ans: Logistic regression is known and used as a linear classifier and hence cannot be used for non-linear data. It is used to come up with a hyper*plane* in feature space to separate observations that belong to a class from all the other observations that do *not* belong to that class. The decision boundary is thus *linear*.

Robust and efficient implementations are readily available (e.g. scikit-learn) to use logistic regression as a linear classifier.

Reasons:

- ✓ the predicted value is continuous, not probabilistic
- ✓ sensitive to imbalance data when using linear regression for classification

---

13. Differentiate between Adaboost and Gradient Boosting.

Ans:

S.No	Adaboost	Gradient Boosting
1	An additive model where shortcomings of previous models are identified by high-weight data points.	An additive model where shortcomings of previous models are identified by the gradient.
2	The trees are usually grown as decision stumps (trees are called as decision stumps)	The trees are grown to a greater depth usually ranging from 8 to 32 terminal nodes.
3	Each classifier has different weights assigned to the final prediction based on its performance.	All classifiers are weighed equally and their predictive capacity is restricted with learning rate to increase accuracy
4	It gives weights to both classifiers and observations thus capturing maximum variance within data.	It builds trees on previous classifier's residuals thus capturing variance in data

5	This model mainly focuses on the prior miscalculated observations and it alters the distribution of dataset	This method trains the learners and focus on reducing the loss function of the week learner
6	The exponential loss provides maximum weights for the samples which are fitted in worse conditions.	Gradient boosting cut down the error components to provide clear explanations and its concepts are easier to adapt and understand
7	Maximum weighted data points are used to identify the shortcomings.	Here, the gradients themselves identify the shortcomings.

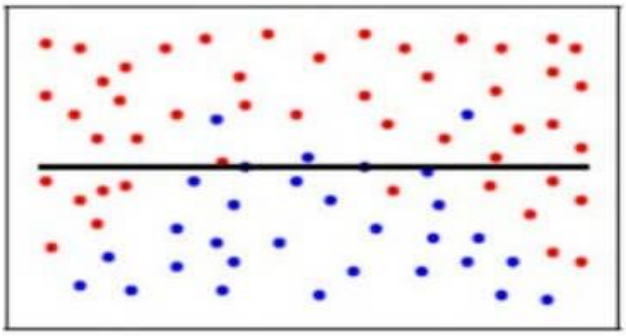
---

14. What is bias-variance trade off in machine learning?

Ans: ***Bias***: Bias is the difference between the Predicted Value and the Expected Value. The model makes certain assumptions when it trains on the data provided. When it is introduced to the testing/validation data, these assumptions may not always to be correct.

In our model, if we use a large number of nearest neighbors, the model can totally decide that some parameters are not important at all.

For example, it can just consider that the Glusoce level and the Blood Pressure decide if the patient has diabetes. This model would make very strong assumptions about the other parameters not affecting the outcome. We can also think of it as a model predicting a simple relationship when the datapoints clearly indicate a more complex relationship:



Mathematically, let the input variables be X and a target variable Y. We map the relationship between the two using a function f.

Therefore,

$$Y = f(X) + e$$

Here ‘e’ is the error that is normally distributed. The aim of our model f'(x) is to predict values as close to f(x) as possible. Here, the Bias of the model is:

$$Bias[f'(X)] = E[f'(X) – f(X)]$$

Variance: Variance is when the model takes into account the fluctuations in the data i.e. the noise as well. So, what happens when our model has a high variance?

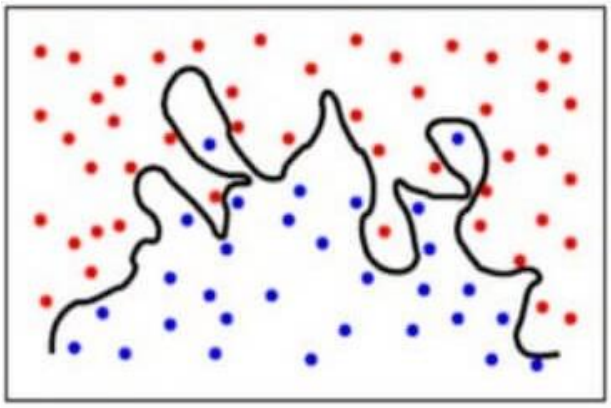
The model will still consider the variance as something to learn from. That is, the model learns too much from the training data, so much that when confronted with new (testing) data, it is unable to predict accurately based on it.

Mathematically, the variance error in the model is:

$$Variance[f(x))=E[X^2]−E[X]^2$$

Since in the case of high variance, the model learns too much from the training data, it is called **overfitting.**

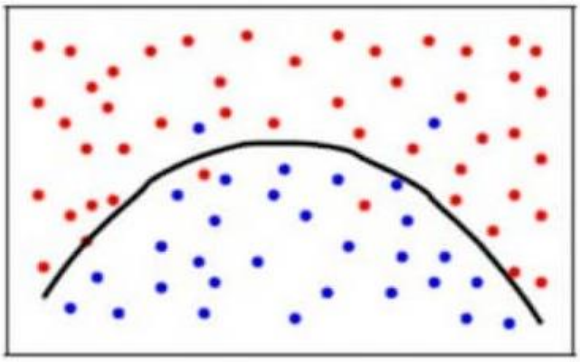
In simple words, the model predicts very complex relationships between the outcome and the input features when a quadratic equation would have sufficed. This is how a classification model would look like when there is a high variance error/when there is **overfitting**:



Conclusion:

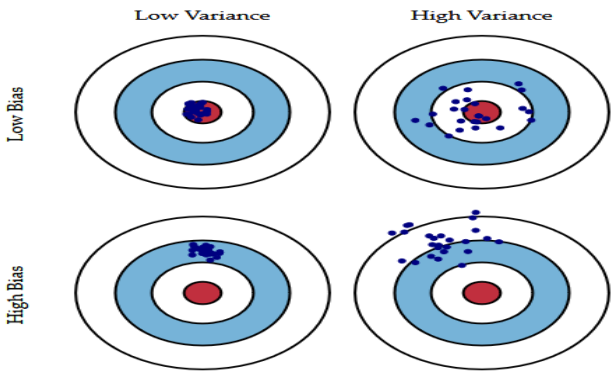
- ✓ A model with a high bias error underfits data and makes very simplistic assumptions on it
- ✓ A model with a high variance error overfits the data and learns too much from it
- ✓ A good model is where both Bias and Variance errors are balanced

Bias-Variance trade off: To achieve a balance between the Bias error and the Variance error, we need a value of **k** such that the model neither learns from the noise (overfit on data) nor makes any assumptions on the data(underfit on data). To keep it simpler, a balanced model would look like this:



The balance between the Bias error and the Variance error is the **Bias-Variance Trade off**.

The following bulls-eye diagram explains the trade off in a much better way:



The centre i.e. the bull's eye is the model result we want to achieve that perfectly predicts all the values correctly. As we move away from the bull's eye, our model starts to make more and more wrong predictions.

A model with low bias and high variance predicts points that are around the center generally, but pretty far away from each other. A model with high bias and low variance is pretty far away from the bull's eye, but since the variance is low, the predicted points are closer to each other.

Conclusion: As we can see that an ideal model would be one where both the bias error and the variance error are low. However, we should always aim for a model where the model score for the training data is as close as possible to the model score for the testing data.

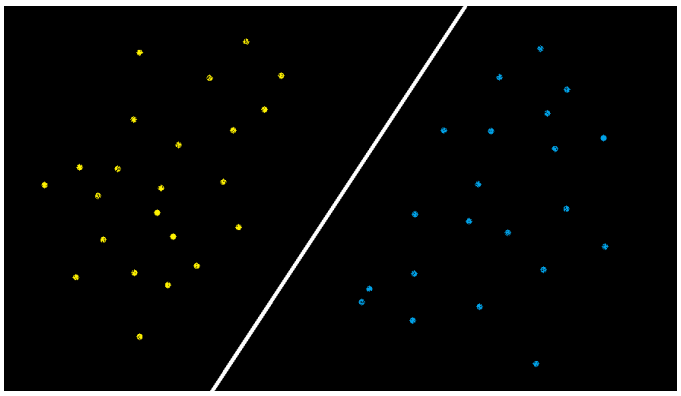
That's where we figured out how to choose a model that is not too complex (High variance and low bias) which would lead to overfitting and nor too simple (High Bias and low variance) which would lead to underfitting.

Bias and Variance plays an important role in deciding which predictive model to use.

---

15. Give short description each of Linear, RBF, Polynomial kernels used in SVM.

Ans: **Linear kernels**: **Linear Kernel** is used when the data is Linearly separable, that is, it can be separated using a single Line. It is one of the most common kernels to be used. It is mostly used when there are a Large number of Features in a particular Data Set.



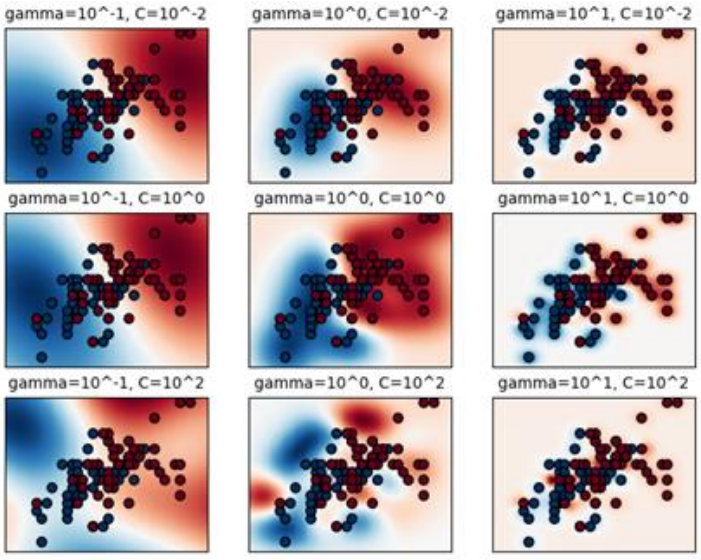
In the above image, there are two set of features "**Blue**" features and the "**Yellow**" Features. Since these can be easily separated or in other words, they are linearly separable, so the Linear Kernel can be used here.

For example, where there are a lot of features, is **Text Classification**, as each alphabet is a new feature. So, we mostly use Linear Kernel in Text Classification.

RBF Kernels: RBF Kernel is popular because of its similarity to K-Nearest Neighbour Algorithm. It has the advantages of K-NN and overcomes the space complexity problem as RBF Kernel Support Vector Machines just needs to store the support vectors during training and not the entire dataset.

The RBF Kernel Support Vector Machines is implemented in the scikit-learn library and has two hyperparameters associated with it, 'C' for SVM and 'γ' for the RBF Kernel. Here, γ is inversely proportional to σ.

$$\gamma \propto \frac{1}{\sigma}$$



From the figure, we can see that as γ increases, i.e., σ reduces, the model tends to overfit for a given value of C. Finding the right γ or σ along with the value of C is essential in order to achieve the best Bias-Variance Trade off.

Polynomial Kernels: A polynomial kernel is a kind of SVM kernel that uses a polynomial function to map the data into a higher-dimensional space. It does this by taking the dot product of the data points in the original space and the polynomial function in the new space.

In a polynomial kernel for SVM, the data is mapped into a higher-dimensional space using a polynomial function. The dot product of the data points in the original space and the polynomial function in the new space is then taken. The polynomial kernel is often used in SVM classification problems where the data is not linearly separable. By mapping the data into a higher-dimensional space, the polynomial kernel can sometimes find a hyperplane that separates the classes.

The polynomial kernel has a number of parameters that can be tuned to improve its performance, including the degree of the polynomial and the coefficient of the polynomial.

For degree d polynomials, the polynomial kernel is defined as:

$$K(x_1, x_2) = (x_1^T x_2 + c)^d$$

where **c** is a constant and **x1** and **x2** are vectors in the original space.