COMP 4353 Data Mining
Extra Assignment
Geraldo Braho
Student ID: 120302835
5/3/2017

Extra Assignment

Item 1. Explain Bias-Variance Tradeoff in predictive models

The aim of any supervised machine learning algorithm is to as precisely as possible estimate the **target function (f)** for the output variable (Y) given the input data (X). The target function is the function that a given supervised machine learning algorithm aims to approximate.

The prediction error for any machine learning algorithm can be broken down into three parts:

- Bias Error
- Variance Error
- Irreducible Error

The irreducible error is the one that can't be reduced no matter which algorithm is used. Because it is the error caused by some unknown variables and interupt the mapping of the input variable to the output one.

Bias are the simplifying assumptions made by a model to make the target function easier to learn.
When less assumptions about the form of target functions are made we have the case of Low Bias. On the other side when more assumtions about the form of target functions are made we then have the High Bias.

Variance is the amount that the estimate of the target function will change if different training data was used. Small changes to the estimate of the target function when changing the dataset means Los Variance, while big changes to the estimate of the target function whenn changing the dataset means High Variance.

Bias-Variance Trade-Off

Above I have said that the goal of any supervised machine learning algorithm is to as precisely as possible estimate the target function for the output data using the given input data. By saying, it means that any supervised machine learning algorithm is aiming for low bias and low variance in order to achieve the good prediction performance.

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

So in order to achieve the best performance, one has to play with the trade-off between these two values, depending on the problem definition and other considerable parameters. [1]

Item 2. Explain Class Imbalance problem in classification problems

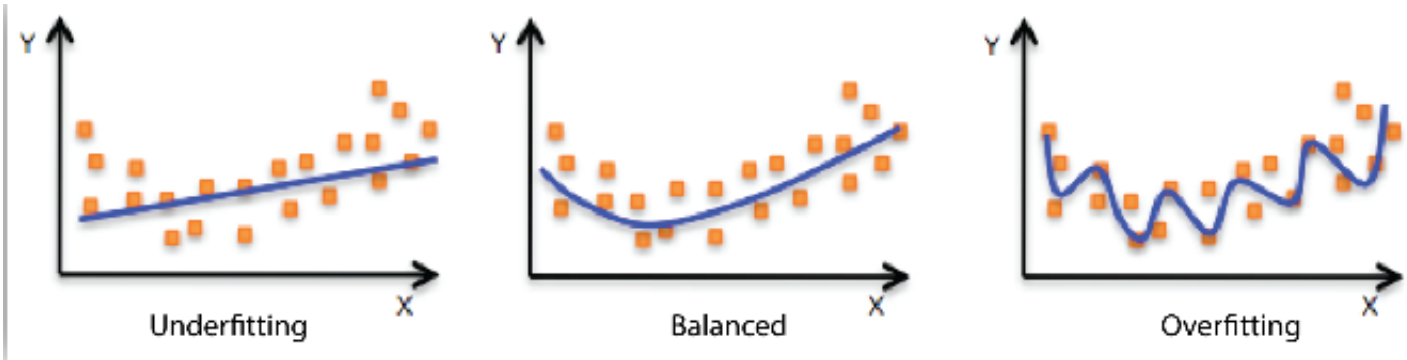
The Class Imbalance Problem is the problem in machine learning meaning that the number of observations of one class is way larger than the number of observations of the other class. This problem is one of the most commmon problems in the machine learning.

The Class Imbalance is a big problem because when the model sees the majority of data instances belong to one class, it will associate any new value with that (majority instances owning) class, without considering the other features necessary for making a correct prediction. There are several possible ways to resolve this issue, we could work on collecting more data, or resampling the current set, generating artifical data based on the existing data set and some other possible approaches you can think of. [2]

Item 3. Explain Overfitting and Underfitting in predictive modeling

Overfitting refers to a model that models the training data too well and it happens when a model learns the detail and noise in the training data to the fact where it negatively impacts the performance of the model on test data. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model. But those concepts don't apply to the new data and give a wrong generalization of the model's ability.

Underfitting refers to a model that can neither model the training data nor generalize to new data. An underfit machine learning model is not a suitable model and will be obvious as it will have poor performance on the training data. Underfitting is often not discussed as it is easy to detect given a good performance metric. The remedy is to move on and try alternate machine learning algorithms.[3]



Item 4. Explain Regularization (methods) and Generalization in predictive modeling

Regularization is tuning or selecting the preferred level of model complexity so the models are better at predicting. If this is not done the models may be too complex and overfit or too simple and underfit, either way giving poor predictions. So basically, regularization helps you to create more generalized model.

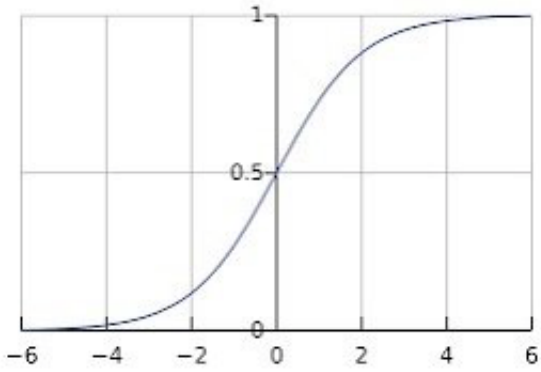
Generalization is the ability of a learning machine to perform accurately on new, unseen examples/tasks. To create good predictive models in machine learning that are capable of generalizing, one needs to know when to stop training the model so that it doesn't overfit.[4]

Item 5. Explain the role of Loss functions in Regression models and give at least 3 examples for regression loss functions; their definitions, formulas, pros and cons

- **Sigmoid Loss**
- **Quadratic Loss**
- **Huber Loss**

Sigmoid Loss function is defined by following characteristics:

- It is S-Shaped and bounded function. It is also called as squashing function, which maps the whole real axis in to finite interval.
  - Usually, the predictions in the classification problem are probability values. So, we don’t want our model to predict the probability value to be below 0 or above 1. Sigmoid function helps to achieve that.
- It is differentiable function.
  - It allows us to apply many popular optimization algorithm to find the optimum point which minimizes the cost function.
- It produces generative model. It model the class conditional probabilities and class priors and use those probabilities to find posterior probability. In simple terms, it model the distribution of individual classes whereas discriminative just model the boundary between the classes
  - Generative model gives a rich representation of relation between features and response. So we can understand the impact of individual features on response.



Quadratic Loss function gives a measure of how accurate a predictive model is. It works by taking the difference between the predicted probability and the actual value – so it is used on classification schemes which produce probabilities (Naive Bayes for example).

The word 'quadratic' means that the highest term in the function is a square. This is used to make sure all the differences are positive. The term 'loss' is self descriptive – it is a measure of the loss of accuracy.

When in use it gives preference to predictors that are able to make the best guess at the true probabilities. It is often used as the criterion of success in probabilistic prediction situations.[15]

The quadratic loss function is most simply expressed by:

$$\sum_j (p_j - a_j)^2$$

Huber Loss is quadratic near zero but linear for large values. It is well suited to errors that are nearly normally distributed with somewhat heavier tails. It is not well suited as a regularization term on the regression coefficients.

Item 6. Explain the role of Loss functions in Classification models and give at least 3 examples for classification loss functions; their definitions, formulas, pros and cons

A loss function measures the discrepancy between the prediction of a machine learning algorithm and the supervised output and represents the cost of being wrong.

- **Hinge Loss**
- **Log Loss**
- **Zero One Loss**

Hinge Loss computes the average distance between the model and the data using hinge loss, a one-sided metric that considers only prediction errors. (Hinge loss is used in maximal margin classifiers such as support vector machines.) If the labels are encoded with +1 and -1, y: is the true value, and w is the predicted decisions as output by decision*function, then the hinge loss is defined as: $$L*\text{Hinge}(y, w) = \max\left{1 - wy, 0\right} = \left|1 - wy\right|_+$$

If there are more than two labels, hinge*loss uses a multiclass variant due to Crammer & Singer. (there is a link for the paper in the scikit learn documentation) and it is defined by: $$L*\text{Hinge}(y_w, y_t) = \max\left{1 + y_t - y_w, 0\right}$$

Log Loss also called logistic regression loss or cross-entropy loss, is defined on probability estimates. It is commonly used in (multinomial) logistic regression and neural networks, as well as in some variants of expectation-maximization, and can be used to evaluate the probability outputs of a classifier instead of its discrete predictions. For binary classification with a true label $y \in \{0, 1\}$ and a probability estimate $p = \Pr(y = 1)$, the log loss per sample is the negative log-likelihood of the classifier given the true label:

$$L_{\log}(y, p) = -\log \Pr(y|p) = -(y \log(p) + (1 - y) \log(1 - p))$$

This extends to the multiclass case as follows. Let the true labels for a set of samples be encoded as a $1 - of - K$ binary indicator matrix Y, i.e., $y_{i,k} = 1$ if sample i has label k taken from a set of K labels. Let P be a matrix of probability estimates, with $p_{i,k} = \Pr(t_{i,k} = 1)$.
Then the log loss of the whole set is:

$$L_{\log}(Y, P) = -\log \Pr(Y|P) = -\frac{1}{N} \sum_{i=0}^{N-1} \sum_{k=0}^{K-1} y_{i,k} \log p_{i,k}$$

To see how this generalizes the binary log loss given above, note that in the binary case, $p_{i,0} = 1 - p_{i,1}$ and $y_{i,0} = 1 - y_{i,1}$, so expanding the inner sum over $y_{i,k} \in \{0, 1\}$ gives the binary log loss.

Zero One Loss computes the sum or the average of the 0-1 classification loss (L_{0-1}) over n_{samples}. By default, the function normalizes over the sample.

In multilabel classification, the zero_one_loss scores a subset as one if its labels strictly match the predictions, and as a zero if there are any errors. By default, the function returns the percentage of imperfectly predicted subsets. To get the count of such subsets instead, set normalize to False.
If $\hat{y_i}$ is the predicted value of the $i - th$ sample and y_i is the corresponding true value, then the 0-1 loss L_{0-1} is defined as:

$$L_{0-1}(y_i, \hat{y_i}) = 1(\hat{y_i} \neq y_i)$$

where $1(x)$ is the indicator function.

[12 - Excuse my bravity for taking the definition from the source refered, but it is not easy to wrap up all these functions with my words.]

Item 7. Explain the Error metrics used in evaluating Classification models and give at least 3 examples for classification error metrics; their definitions, formulas, pros and cons

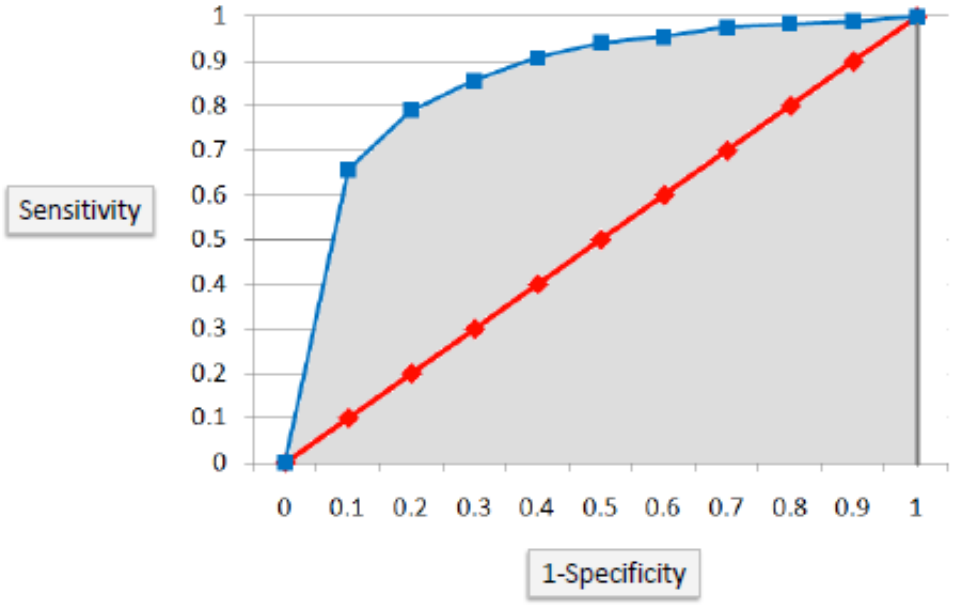
- **Confusion Matrix**
- **Area Under the ROC curve (AUC – ROC)**
- **Gain and Lift charts**

Confusion Matrix shows the number of correct and incorrect predictions made by the classification model compared to the actual outcomes (target value) in the data. The matrix is NxN, where N is the number of target values (classes). Performance of such models is commonly evaluated using the data in the matrix. The following table displays a 2x2 confusion matrix for two classes (Positive and Negative).

Confusion Matrix		Target			
		Positive	Negative		
Model	Positive	a	b	*Positive Predictive Value*	a/(a+b)
	Negative	c	d	*Negative Predictive Value*	d/(c+d)
		Sensitivity	*Specificity*	**Accuracy** = (a+d)/(a+b+c+d)	
		a/(a+c)	d/(b+d)		

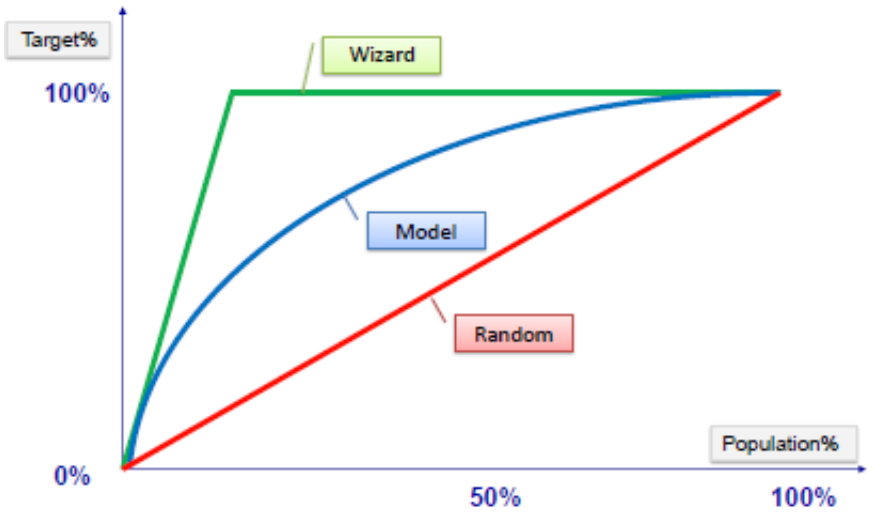
Accuracy : the proportion of the total number of predictions that were correct.
Positive Predictive Value or Precision : the proportion of positive cases that were correctly identified.
Negative Predictive Value : the proportion of negative cases that were correctly identified.
Sensitivity or Recall : the proportion of actual positive cases which are correctly identified.
Specificity : the proportion of actual negative cases which are correctly identified.

Area Under the ROC curve (AUC – ROC) is often used as a measure of quality of the classification models. A random classifier has an area under the curve of 0.5, while AUC for a perfect classifier is equal to 1. In practice, most of the classification models have an AUC between 0.5 and 1.

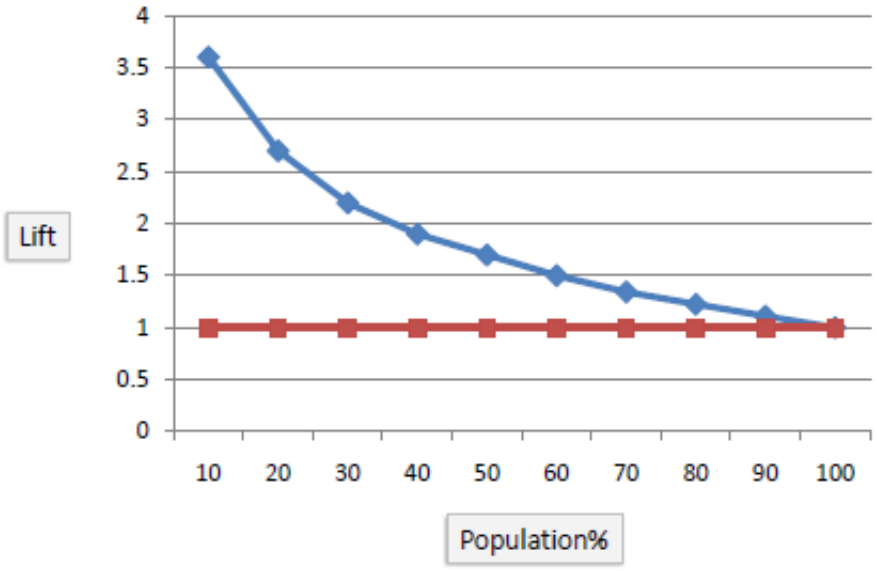


An area under the ROC curve of 0.8, for example, means that a randomly selected case from the group with the target equals 1 has a score larger than that for a randomly chosen case from the group with the target equals 0 in 80% of the time. When a classifier cannot distinguish between the two groups, the area will be equal to 0.5 (the ROC curve will coincide with the diagonal). When there is a perfect separation of the two groups, which means no overlapping of the distributions, the area under the ROC curve reaches to 1 (the ROC curve will reach the upper left corner of the plot).

Gain and Lift charts are a measure of the effectiveness of a classification model calculated as the ratio between the results obtained with and without the model. Gain and lift charts are visual aids for evaluating performance of classification models. However, in contrast to the confusion matrix that evaluates models on the whole population gain or lift chart evaluates model performance in a portion of the population.



The lift chart shows how much more likely we are to receive positive responses than if we contact a random sample of customers. For example, by contacting only 10% of customers based on the predictive model we will reach 3 times as many respondents, as if we use no model.



[13][14]

Item 8. Explain the Error metrics used in evaluating Regression models and give at least 3 examples for regression error metrics; their definitions, formulas, pros and cons

An Error Metric is a metric used to measure the error of a forecasting model. They can provide a way for forecasters to quantitatively compare the performance of competing models. Some of the common error metrics used in evaluating regression models are:

- **Mean Squared Error (RMSE)**
- **Median Absolute Error**
- **R^2 score, the coefficient of determination**

Mean Squared Error (MSE) is the most popular evaluation metric used in regression problems. It follows an assumption that error are unbiased and follow a normal distribution. The key points to consider on MSE:

- The power of 'square root' empowers this metric to show large number deviations.
- The 'squared' nature of this metric helps to deliver more robust results which prevents cancelling the positive and negative error values. In other words, this metric aptly displays the plausible magnitude of error term.
- It avoids the use of absolute error values which is highly undesirable in mathematical calculations.

- When there are more samples, reconstructing the error distribution using MSE is considered to be more reliable.
- RMSE is highly affected by outlier values. Hence, it is good to make sure that outliers are removed from your data set prior to using this metric.
- As compared to mean absolute error, RMSE gives higher weightage and punishes large errors.

MSE metric is given by:

$$\text{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y_i})^2.$$

Usage of mean squared error with scikit library:

```
from sklearn.metrics import mean_squared_error
y_true = [3, -0.5, 2, 7]
y_pred = [2.5, 0.0, 2, 8]
mean_squared_error(y_true, y_pred)
```

Median Absolute Error is particularly interesting because it is robust to outliers. The loss is calculated by taking the median of all absolute differences between the target and the prediction. If $\hat{y_i}$ is the predicted value of the $i - th$ sample and y_i is the corresponding true value, then the median absolute error (MedAE) estimated over n_{samples} is defined as:

$$\text{MedAE}(y, \hat{y}) = \text{median}(| y_1 - \hat{y_1} |, \ldots, | y_n - \hat{y_n} |).$$

Simple usage of the median_absolute_error function with scikit learn library:

```
from sklearn.metrics import median_absolute_error
y_true = [3, -0.5, 2, 7]
y_pred = [2.5, 0.0, 2, 8]
median_absolute_error(y_true, y_pred)
```

R² score, the coefficient of determination - The r2_score function computes R², the coefficient of determination. It provides a measure of how well future samples are likely to be predicted by the model. Best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y, disregarding the input features, would get a R^2 score of 0.0. If $\hat{y_i}$ is the predicted value of the $i - th$ sample and y_i is the corresponding true value, then the score R² estimated over n_{samples} is defined as

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y_i})^2}{\sum_{i=0}^{n_{\text{samples}}-1} (y_i - \bar{y})^2}$$

where $\bar{y} = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} y_i$.

Usage example within scikit learn library:

```
from sklearn.metrics import r2_score
y_true = [3, -0.5, 2, 7]
y_pred = [2.5, 0.0, 2, 8]
r2_score(y_true, y_pred)
```

[12]

Item 9. Explain the Distance metrics used in similarity of data instances and give at least 3 examples for distance metrics; their definitions, formulas, pros and cons

The similarity measure is the measure of how much alike two data objects are. Similarity measure in a data mining context is a distance with dimensions representing features of the objects. If this distance is small, it will be the high degree of similarity where large distance will be the low degree of similarity.

The similarity is subjective and is highly dependent on the domain and application. Care should be taken when calculating distance across dimensions/features that are unrelated. The relative values of each element must be normalized, or one feature could end up dominating the distance calculation. Similarity are measured in the range 0 to 1 [0,1].

Two main consideration about similarity:
Similarity = 1 if X = Y (Where X, Y are two objects) Similarity = 0 if X ≠ Y

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

Minkowski $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

Euclidean distance: is the most common use of distance. In most cases when people said about distance, they will refer to Euclidean distance. Euclidean distance is also known as simply distance. When data is dense or continuous, this is the best proximity measure.

The Euclidean distance between two points is the length of the path connecting them.The Pythagorean theorem gives this distance between two points.

Euclidean distance implementation in python:

```
from math import*

def euclidean_distance(x,y):

    return sqrt(sum(pow(a-b,2) for a, b in zip(x, y)))

print euclidean_distance([0,3,4,5],[7,6,3,-1])
```

Manhattan distance: is a metric in which the distance between two points is the sum of the absolute differences of their Cartesian coordinates. In a simple way of saying it is the total sum of the difference between the x coordinates and y-coordinates.

Suppose we have two points A and B if we want to find the Manhattan distance between them, just we have, to sum up, the absolute x-axis and y – axis variation means we have to find how these two points A and B are varying in X-axis and Y- axis. In a more mathematical way of saying Manhattan distance between two points measured along axes at right angles.

In a plane with p1 at (x1, y1) and p2 at (x2, y2).

Manhattan distance = $|x1 – x2| + |y1 – y2|$

This Manhattan distance metric is also known as Manhattan length, rectilinear distance, L1 distance or L1 norm, city block distance, Minkowski's L1 distance, taxi-cab metric, or city block distance.

Manhattan distance implementation in python:

```
from math import*
def manhattan_distance(x,y):

    return sum(abs(a-b) for a,b in zip(x,y))

print manhattan_distance([10,20,10],[10,20,20])
```

Minkowski distance: is a generalized metric form of Euclidean distance and Manhattan distance.

The way distances are measured by the Minkowski metric of different orders between two objects with three variables (In the image it displayed in a coordinate system with x, y ,z-axes).

Synonyms of Minkowski: Different names for the Minkowski distance or Minkowski metric arise from the order:

λ = 1 is the Manhattan distance. Synonyms are L1-Norm, Taxicab or City-Block distance. For two vectors of ranked ordinal variables, the Manhattan distance is sometimes called Foot-ruler distance.
λ = 2 is the Euclidean distance. Synonyms are L2-Norm or Ruler distance. For two vectors of ranked ordinal variables, the Euclidean distance is sometimes called Spear-man distance.
λ = ∞ is the Chebyshev distance. Synonyms are Lmax-Norm or Chessboard distance. reference.

Minkowski distance implementation in python:

```
#!/usr/bin/env python


from math import*
from decimal import Decimal

def nth_root(value, n_root):

    root_value = 1/float(n_root)
    return round (Decimal(value) ** Decimal(root_value),3)

def minkowski_distance(x,y,p_value):

    return nth_root(sum(pow(abs(a-b),p_value) for a,b in zip(x, y)),p_valu
e)

print minkowski_distance([0,3,4,5],[7,6,3,-1],3)
```

[11]

Item 10. Explain the Accuracy and Precision in data quality

Data accuracy is one of the components of data quality. It refers to whether the data values stored for an object are the correct values. A data values must be the right value and must be represented in a consistent and unambiguous form.

There are two characteristics of accuracy: form and content. Form is important because it eliminates ambiguities about the content. The birth date example is ambiguous because the reviewer would not know whether the date was invalid or just erroneously represented. You cannot tell the representation from the value and thus need discipline in creating the date values in order to be accurate. A value is not accurate if the user of the value cannot tell what it is.

Precision is the depth of knowledge encoded in data. A summary report has low precision, while a detailed spreadsheet has high precision. Data with high precision can be used by specialists such as developers, designers, and statisticians to write articles that explain the data, to create infographics, or to transform the same information into other consumable forms. Data precision is refered as the closeness between all possible interpretations of a data object.[8]

Item 11. Explain the Variable Transformation in data processing

In data processing transformation is the replacement of a variable by a simple formulation or a function of that variable: for example, replacing a variable x by the square root of x or the logarithm of x. In a stronger sense, a transformation is a replacement that changes the shape of a distribution or relationship.

For categorical to numerical scales, we have to assign an appropriate numerical number to a categorical value according to needs. Categorical variables can be ordinal (such as less, moderate, and strong) and nominal (such as red, yellow, blue, and green). For example, a binary variable {yes, no} can be transformed into “1 = yes and 0 = no.” Note that transforming a numerical value to an ordial value means transformation with order, while transforming to a nominal value is a less rigid transformation.There is no unique procedure and the only criterion is to transform the data for convenience of use during the data mining stage.[7]

Item 12. Explain the Validation Methods in evaluating predictive models and give at least 3 examples for validation methods; their definitions, formulas, pros and cons

There are three types of validations, they are as following:

Apparent: The performance on sample used to develop model as per the protocol of the respective study. In this apparent method: It is very easy to calculate and the results in optimistic performance estimates.

Internal: The performance on population underlying the sample of the particular study. It is more difficult to calculate one ad the test model in new data, random from underlying population.

External: The performance on related but slightly different population in the study. It is the moderately easy to calculate when new data are available in the prediction model. The test model in new data, different from development population.[10]

Item 13. Explain Scaling methods used in data preprocessing and give at least 3 examples for scaling methods; their definitions, formulas, pros and cons

The result of standardization (or Z-score normalization) is that the features will be rescaled so that they'll have the properties of a standard normal distribution with μ=0 and σ=1 where μ is the mean (average) and σ is the standard deviation from the mean.

MinMaxScaler:
Transforms features by scaling each feature to a given range. Scales and translates each feature individually such that it is in the given range on the training set, so it is between zero and one. The transformation is given by:
X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0)) X_scaled = X_std * (max - min) + min
where min, max = feature_range.
This transformation is often used as an alternative to zero mean, unit variance scaling.

StandardScaler:
Standardize features by removing the mean and scaling to unit variance.
Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. Mean and standard deviation are then stored to be used on later data using the transform method. Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual feature do not more or less look like standard normally distributed data (e.g. Gaussian with 0 mean and unit variance). For instance many elements used in the objective function of a learning algorithm (such as the RBF kernel of Support Vector Machines or the L1 and L2 regularizers of linear models) assume that all features are centered around 0 and have variance in the same order. If a feature has a variance that is orders of magnitude larger that others, it might dominate the objective function and make the estimator unable to learn from other features correctly as expected.

MaxAbsScaler:
Scale each feature by its maximum absolute value. This estimator scales and translates each feature individually such that the maximal absolute value of each feature in the training set will be 1.0. It does not shift/center the data, and thus does not destroy any sparsity.[9]

Item 14. Explain Ensemble modeling methods (bagging, boosting and random forest); their definitions, formulas, pros and cons

Bagging and **Boosting** are general strategies for improving classifier and preictor accuracy. They use combination of models.

Bagging - is the way to decrease the variance of the prediction by generating additional data for training from the original dataset using combinations with repetitions to produce multisets of the same cardinality/size as the original data. By increasing the size of the training set, the model predictive force can't be improved, but just decrease the variance, narrowly tuning the prediction to expected outcome.

Boosting - is a two-step approach, where one first uses subsets of the original data to produce a series of averagely performing models and then "boosts" their performance by combining them together using a particular cost function (=majority vote). Unlike bagging, in the classical boosting the subset creation is not random and depends upon the performance of the previous models: every new subsets contains the elements that were (likely to be) misclassified by previous models.

Random Forest is an improvement over bagged decision trees. And it operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training sets. [5][6]

Item 15. Explain Gain Ratio and Information Gain used in Decision Tree modeling

Gain Ratio - a modification of the information gain that reduces its bias. Gain Ratio takes number and size of branches into account when choosing an attribute. It corrects the information gain by taking the intrinsic information of a split into account. The Gain Ratio is defined as: **Gain(A)/SplitInfo(A)** Where split information value represents the potential information generated by splitting the training data set into partitions, corresponding to the outcomes on attribute A. The attribute with the maximum gain ratio is selected as the splitting attribute.

Information Gain - tells us how important a given attribute of the feature vectors is. The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).[5]

References:

[1] Gentle Introduction to the Bias-Variance Trade-Off in Machine Learning by Jason Brownlee
[2] Class Imbalance Problem - Log0 Github Repository and Blog
[3] Overfitting and Underfitting With Machine Learning Algorithms by Jason Brownlee
[4] What is the difference between regularization and generalization? by Quora answers
[5] Data Mining Concepts and Technoques (Jiawei Han and Micheline Kamber)
[6] Bagging anf Random Forest Ensemble Algorithms by Jason Brownlee
[7] Transformations: an introduction [http://fmwww.bc.edu/repec/bocode/t/transint.html (http://fmwww.bc.edu/repec/bocode/t/transint.html)]
[8] Open Government Data: The Book [Data Quality: Precision, Accuracy and Cost]
[9] Processing Data by Scikit Learn
[10] What are the different ways of performing validation of prediction model? by Research Gate answers
[11] Five most popular similarity measures implementation in python by Saimadhu Polamuri
[12] Model evaluation: quantifying the quality of predictions by Scikit Learn library
[13] 7 Important Model Evaluation Error Metrics Everyone should know by Tavish Srivastava
[14] An introduction to data mining by Saed Sayad
[15] Quadratic Loss Function by Butler Blog Post

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: