

1. Vad är data leakage, varför är det dåligt och hur kan du undvika data leakage?

In [statistics](#) and [machine learning](#), **leakage** (also known as **data leakage** or **target leakage**) is the use of [information](#) in the model training process which would not be expected to be available at [prediction](#) time, causing the predictive scores (metrics) to overestimate the model's utility when run in a production environment.

Leakage is often subtle and indirect, making it hard to detect and eliminate. Leakage can cause a statistician or modeler to select a suboptimal model, which could be outperformed by a leakage-free model.

Leakage can occur in many steps in the machine learning process. The leakage causes can be sub-classified into two possible sources of leakage for a model: features and training examples.^[1]

Feature leakage^[edit]

Feature or column-wise leakage is caused by the inclusion of columns which are one of the following: a duplicate label, a proxy for the label, or the label itself. These features, known as anachronisms, will not be available when the model is used for predictions, and result in leakage if included when the model is trained.^[2]

For example, including a "MonthlySalary" column when predicting "YearlySalary"; or "MinutesLate" when predicting "IsLate"; or more subtly "NumOfLatePayments" when predicting "ShouldGiveLoan".

Training example leakage^[edit]

Row-wise leakage is caused by improper sharing of information between rows of data. Types of row-wise leakage include:

- Premature [featurization](#); leaking from premature featurization before [CV](#)/Train/Test split (must fit MinMax/ngrams/etc on only the train split, then transform the test set)
- Duplicate rows between train/validation/test (e.g. oversampling a dataset to pad its size before splitting; e.g. different rotations/augmentations of a single image; [bootstrap sampling](#) before splitting; or duplicating rows to [up sample](#) the minority class)
- [Non-i.i.d.](#) data
 - Time leakage (e.g. splitting a time-series dataset randomly instead of newer data in test set using a TrainTest split or rolling-origin cross validation)
 - Group leakage -- not including a grouping split column (e.g. [Andrew Ng](#)'s group had 100k x-rays of 30k patients, meaning ~3 images per patient. The paper used random splitting instead of ensuring that all images of a patient was in the same split. Hence the model partially memorized the patients instead of learning to recognize pneumonia in chest x-rays. Revised paper had a drop in scores.

Scaling training data, test data to the training data, to avoid data leakage.

2. Beskriv vad poängen kan vara att dela upp datasetet till träningsdata och testdata för maskininlärning?

The test set should be exclusive to the training set. In other words, the test samples should not appear in the training set as much as possible, and should not be used in the training set. The simplest way to split the modelling dataset into training and testing sets is to assign 2/3 data points to the former and the remaining one-third to the latter.

3. Beskriv vad poängen kan vara att dela upp datasetet till tränings-, validerings- och testdataset för maskininlärning?

- For small-scale sample sets, a common ratio is 2/3 training set, 1/6 validation set, and 1/6 test set.
- For large-scale sample sets (above millions) : as long as the number of validation sets and test sets is sufficient
- If hyperparameters is small or easy to tune, can reduce the proportion of the validation set and leave more space to the training set.

4. Hur skiljer sig batch gradient descent och mini batch gradient descent?

Batch gradient descent: uses the whole training batch at each iteration. Drawbacks: 1. Slow on large training set. 2. Get stuck on a local minima, instead of global minima.

Gradient descent

Stochastic gradient descent

Batch gradient descent is slow on large amount of data as the whole training set is used on computing the gradient at each iteration. Also a drawback is that it might get stuck on a local minima instead of a global.

Another way and more general way to train linear regression is by using gradient descent, which is an iterative optimization approach. It works by taking a cost function e.g. MSE,

$$C = \frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2 = \frac{1}{m} \sum_{i=1}^m (y_i - \vec{\theta} \cdot \vec{x}_i)^2$$

Here $\vec{\theta}$ are the parameters that is weights and bias or $\hat{\beta}$ when referred to last lecture.

The main idea with gradient descent is to minimize the cost function (a number representing the error, between label and prediction) through taking small steps one at a time in the direction of the steepest descent. Through multivariate calculus the steepest ascent is the gradient of the multivariate function, so we take the negative direction of the gradient.

$$\vec{\theta}_{j+1} = \vec{\theta}_j - \eta \nabla_{\vec{\theta}} C(\vec{\theta})$$

where η is the learning rate, $\nabla_{\vec{\theta}}$ is the gradient w.r.t. $\vec{\theta}$ and j is the iteration step. By adjusting η we get different step sizes. With some calculus the gradient becomes:

$$\nabla_{\vec{\theta}} C(\vec{\theta}) = \dots = \frac{2}{m} X^T (X\vec{\theta} - \vec{y})$$

This is called batch gradient descent as it uses whole training batch, which will be slow on large training set.

Stochastic gradient descent (SGD) instead calculates gradient based on one instance, which causes the road to minimum to be very bumpy. Due to its stochasticity it has higher chance to jump out of local minima. It can come close to but never reach the optimal solution.

Mini batch gradient descent

Computes gradient on small batches of randomly chosen training data. It will end up closer to the minimum than SGD. The big advantage is performance when using GPUs (graphics processing unit).

5. Vad är en aktiveringsfunktion i en artificiell neuron? Ge två exempel på aktiveringsfunktioner.

instances of neural networks, the *sigmoid* activation function was favored, sigmoid

$$g(z) = \frac{e^z}{1 + e^z} = \frac{1}{1 + e^{-z}}, \quad (10.4)$$

which is the same function used in logistic regression to convert a linear function into probabilities between zero and one (see Figure 10.2). The preferred choice in modern neural networks is the *ReLU (rectified linear unit)* activation function, which takes the form

ReLU
rectified
linear unit

$$g(z) = (z)_+ = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{otherwise.} \end{cases} \quad (10.5)$$

A ReLU activation can be computed and stored more efficiently than a sigmoid activation. Although it thresholds at zero, because we apply it to a linear function (10.2) the constant term w_{k0} will shift this inflection point.

So in words, the model depicted in Figure 10.1 derives five new features by computing five different linear combinations of X , and then squashes each through an activation function $g(\cdot)$ to transform it. The final model is linear in these derived variables.

The name *neural network* originally derived from thinking of these hidden units as analogous to neurons in the brain — values of the activations $A_k = h_k(X)$ close to one are *firing*, while those close to zero are *silent* (using the sigmoid activation function).

In [artificial neural networks](#), the **activation function** of a node defines the output of that node given an input or set of inputs. only *nonlinear* activation functions allow such networks to compute nontrivial problems using only a small number of nodes, and such activation functions are called **nonlinearities**

Sigmoid / Logistic Activation Function :

$$f(x) = \frac{1}{1 + e^{-x}}$$

Binary Step Function:

Binary step

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$$

6. Du har ett program som klassificerar email till spam eller hams (inte spam). Vilka evalueringsmetrics ska du kolla på och varför?

We want to have high precision: because we don't want to have any hams in spam box.

We can accept lower recall: It is Ok if we get spams in inbox.

Best: High precision and high recall.

7. Ett brandlarm larmar ofta när det inte brinner. Tror du precision är högt eller lågt, förklara varför.

Precision är lågt. Because the False Positive is large and $\text{precision} = \text{TP} / (\text{TP} + \text{FP})$

8. Förklara vad curse of dimensionality är och hur det kan påverka en maskininlärningsmodell.

However, spreading 50 observations over $p = 20$ dimensions results in a phenomenon in which a given observation has no nearby neighbors—this is the so-called curse of dimensionality. That is, curse of dimensionality the K observations that are nearest to a given test observation x_0 may be very far away from x_0 in p -dimensional space when p is large, leading to a very poor prediction of $f(x_0)$ and hence a poor KNN fit. As a general rule, parametric methods will tend to outperform non-parametric approaches when there is a small number of observations per predictor.

When the number of features p is large, there tends to be a deterioration in the performance of KNN and other local approaches that perform prediction using only observations that are near the test observation for which a prediction must be made. This phenomenon is known as the curse of dimensionality, and it ties into the fact that curse of dimensionality non-parametric approaches often perform poorly when p is large.

9. Vad är skillnaden på feature standardization och normalization?

Standardization:
$$X' = \frac{X - \mu}{\sigma}$$
 X' (mean) = 0, $\sigma=1$

Normalization :
$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$
 X' is between 0 and 1

10. Du har ett dataset som ser ut som följande (1000 observationer):

temperatur i °C	vind (m/s)
22.5	2.0
8.3	10.3
13.2	4
...	...

En meteorolog vill gruppera dessa. Ge ett förslag på hur hen skulle kunna göra med en maskininlärningsalgoritm.

Make dummy variables for temperatures: 1. ($X < -20$), 2. ($-20 \leq X < -10$), ..., 10 degrees every group...

You can make a simple linear regression model for X and y (vind).

Or you can dummy the vind(y) to 0 and 1 (vind>10: 1; vind< 10 : 0) to draw a logistic regression.

11. Många maskininlärningsmodeller har svårt att arbeta med textdata direkt. Ge förslag på hur man skulle kunna förbearbeta en textsträng.

Term frequency - inverse document frequency (TF-IDF)

TF-IDF is a way to represent how important a word is across a corpus of documents. Basically it is a vector with numeric weights on each word, where higher weights is put on rarer terms.

Term frequency

term frequency $tf(t, d)$ - relative frequency of term t in document d , i.e. how frequent a term occurs in a document

$$tf(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

, where $f_{t,d}$ the raw count, is the amount of time the term t is in document d . The denominator is the total number of terms in the document. Also $tf(t, d)$ could be defined in several ways, and a simple way is to just equate it to the raw frequency count.

Lec11-NLP_intro.ipynb

Lectures > Lec11-NLP_intro.ipynb > **TF-IDF** for interacting with the code

Code + Markdown Run All Clear Outputs of All Cells Restart Interrupt Variables Outline MaskinInlarning-Yuna-Liu-sgUjmrCu (Python 3.9.6)

TF-IDF

- Term frequency - inverse document frequency
- TF-IDF is a way to represent how important a word is across a corpus of documents. Basically it is a vector with numeric weights on each word, where higher weights is put on rarer terms.

The inverse document frequency $idf(t, D)$ gives information on the rarity of the word in all documents D .

$$idf(t, D) = \log \frac{|D|}{1 + |\{d \in D : t \in d\}|}$$

, where $|D|$ is the number of documents in the corpus, $1 + |\{d \in D : t \in d\}|$ is the number of documents where the word t occurs, we add 1 to avoid division by zero in case the word is not in the corpus.

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D)$$

Feature extraction with sklearn

- CountVectorizer - creates a bag of words model

Jupyter Server: local Cell 1 of 24 下午 4:22 6/4/2022

Lec11-NLP_intro.ipynb

Lectures > Lec11-NLP_intro.ipynb > **Feature extraction with sklearn** for interacting with the code

Code + Markdown Run All Clear Outputs of All Cells Restart Interrupt Variables Outline MaskinInlarning-Yuna-Liu-sgUjmrCu (Python 3.9.6)

Feature extraction with sklearn

- CountVectorizer - creates a bag of words model
- TfidfTransformer - transforms it using TF-IDF
- TfidfVectorizer - does CountVectorizer and TfidfTransformer

```
1 from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer, TfidfVectorizer
2
3 count_vectorizer = CountVectorizer()
4 # CountVectorizer() is a class, which we should initialize
5 bag_of_words_sparse = count_vectorizer.fit_transform([review1, review2])
6 # we often work with sparse matrix when deal with text data
7 bag_of_words_sparse.todense(), count_vectorizer.get_feature_names_out()
8 # count_vectorizer.get_feature_names() works in older version of sklearn 0.24.2
```

[13]

```
(matrix([[1, 1, 2, 0, 0, 1, 0],
         [0, 1, 0, 1, 1, 1, 1]], dtype=int64),
 array(['about', 'book', 'love', 'no', 'okay', 'this', 'was'], dtype=object))
```

```
1 # note that it ignores one letter words such as "I"
2 pd.DataFrame(bag_of_words_sparse.todense(), columns = count_vectorizer.get_feature_names_out())
```

Jupyter Server: local Cell 1 of 24 下午 4:24 6/4/2022

12. Du ska Ananda KNN för klassificering. Ge förslag på hur du skulle kunna gå tillväga för att bestämma ett lämpligt värde på antalet neighbors.

KNN

KNN or k-nearest neighbours is a supervised machine learning algorithm that can be used for both regression or classification. It calculates the distance between

a test data point and all training data, find k training points nearest to the test data. Then it does majority voting to classify that test point to majority of the class of the training data points that are closest. For regression instead it takes an average of those k points that are closest.

In KNN it is absolute necessity to do feature scaling as the distance calculated using a distance metric can be very wrong if the features are in different scales.

Elbow plot to choose k

Note that this is actually cheating as we use the testing data for hyperparameter tuning. This gives us data leakage. A correct way to do it here is to split the data set into train|validation|test sets, perform elbow plot on validation data. Choose a k and do the test on testing data. As our data set is so small, a better way would be to use cross-validation which I will show in the next lecture. Here I will do the cheating method to show the elbow plot

We split the data into train|validation|test sets, and then perform elbow plot on validation data. We choose the k from min value of errors (`error = 1-accuracy_score(y_test, y_pred)`)

13. Beskriv begreppen: perceptron, multilayered perceptron, feedforward network, fully connected layers.

A Perceptron is a neural network unit that does certain computations to detect features or business intelligence in the input data.

A multi-layer perceptron has one input layer and for each input, there is one neuron(or node), it has one output layer with a single node for each output and it can have any number of hidden layers and each hidden layer can have any number of nodes.

A feed-forward neural network is a biologically inspired classification algorithm. It consists of a number of simple neuron-like processing units, organized in layers and every unit in a layer is connected with all the units in the previous layer.

Fully Connected layers in a neural networks are those layers where all the inputs from one layer are connected to every activation unit of the next layer.

14. Aktiveringsfunktionen rectified linear unit (ReLU) är definierad som $f(z) = \max(0, z)$. En nod i ett MLP har denna aktiveringsfunktion och får inputs $\vec{x} = (1, 2, 3)^T$ och har vikterna $\vec{w} = (0, 2, 1)^T$ och bias $b = -2$. Beräkna output y för denna nod.

$$\vec{w}^T \vec{x} + b = (0, 2, 1) \cdot \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} - 2$$

$$= 0 + 4 + 3 - 2 = 5$$

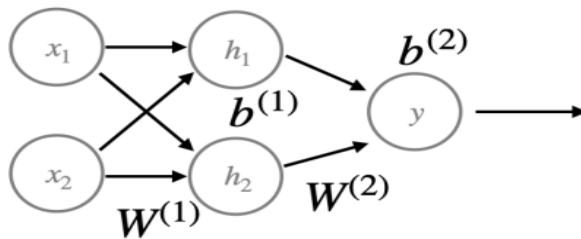
$$f(z) = \max(0, z), \quad z = 5 \text{ (output)}$$

15. XOR grind i digitalteknik är en exklusiv disjunktion vilket innebär:

A	B	A XOR B
1	1	0
1	0	1
0	1	1
0	0	0

Visa att följande MLP kan simulera XOR-grind. Varje nod är en perceptron, dvs att aktiveringsfunktionen är en stegfunktion.

$$W^{(1)} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, W^{(2)} = \begin{pmatrix} 1 \\ -2 \end{pmatrix}, b^{(1)} = \begin{pmatrix} 0 \\ -1 \end{pmatrix}, b^{(2)} = 0$$



$$\vec{w}_1^T \vec{x} + b_1 = \begin{pmatrix} 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 0 \\ -1 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \end{pmatrix} + \begin{pmatrix} 0 \\ -1 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

$$\vec{w}_2^T \vec{x} + b_2 = \begin{pmatrix} 1 & -2 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 1 \end{pmatrix} + 0 = 2 - 2 + 0 = 0 \quad (\text{A XOR B})$$

The same result when $\vec{x} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \Rightarrow 1$

16. Formeln för Naive Bayes är:

$$y = \arg \max_{k \in 1, \dots, K} P(C_k) \prod_{i=1}^n P(x_i | C_k)$$

Du har ett dataset som består 100 reviews, varav 80 var positiva och 20 negativa. Nedan finns tabeller på hur frekvent olika ord förekommer för positiva respektive negativa reviews

Positiva reviews:

	happy	good	book	author	bad
Antal	50	60	20	10	1

Negativa reviews:

	happy	good	book	author	bad
Antal	2	5	40	40	40

Vi får ett review med texten "happy bad", använd Naive Bayes och klassificera den till positiv eller negativ review.

$$80 \quad C_1 \quad \boxed{\text{Positive}} : P(C_1) = \frac{80}{100} = \frac{4}{5}$$

$$20 \quad C_2 \quad \boxed{\text{Negative}} : P(C_2) = \frac{20}{100} = \frac{1}{5}$$

$$P(x_i|C_1) :$$

happy	good	book	author	bad
50	60	20	10	1
$\frac{50}{141}$	$\frac{60}{141}$	$\frac{20}{141}$	$\frac{10}{141}$	$\frac{1}{141}$

$$P(x_i|C_2) :$$

happy	good	book	author	bad
2	5	40	40	40
$\frac{2}{127}$	$\frac{5}{127}$	$\frac{40}{127}$	$\frac{40}{127}$	$\frac{40}{127}$

New words: ("happy bad")

$$\begin{aligned} \text{Positive} &= P(\text{positive}) \cdot P(\text{happy}|\text{positive}) \cdot P(\text{bad}|\text{positive}) \\ &= \frac{4}{5} \cdot \frac{50}{141} \cdot \frac{1}{141} \approx 0,002 = 0,2\% \end{aligned}$$

$$\begin{aligned} \text{Negative} &= P(\text{Negative}) \cdot P(\text{happy}|\text{negative}) \cdot P(\text{bad}|\text{negative}) \\ &= \frac{1}{5} \cdot \frac{2}{127} \cdot \frac{40}{127} \approx \frac{0,001}{0,001} = 0,1\% \end{aligned}$$

17. Beskriv kort skillnaderna mellan decision tree och random forest.

The difference is that decision trees are graphs that illustrate all possible outcomes of a decision using a branching approach. In contrast, the random forest algorithm output are a set of decision trees that work according to the output.

Decision tree regression

The goal is to stratify or segment the players into several regions. In decision tree for regression, the algorithm creates a tree to minimize the RSS (residual sum of squares). The tree-building process uses recursive binary splitting, a top-down greedy approach to divide the predictor space into branches. For example the baseball dataset with years and hits we could have a split into the following regions:

$$R_1 = \{X|Years < 4.5\}, R_2 = \{X|Years \geq 4.5, Hits < 117.5\}, R_3 = \{X|Years \geq 4.5, Hits \geq 117.5\}$$

For decision trees, you shouldn't scale the data.

Bagging

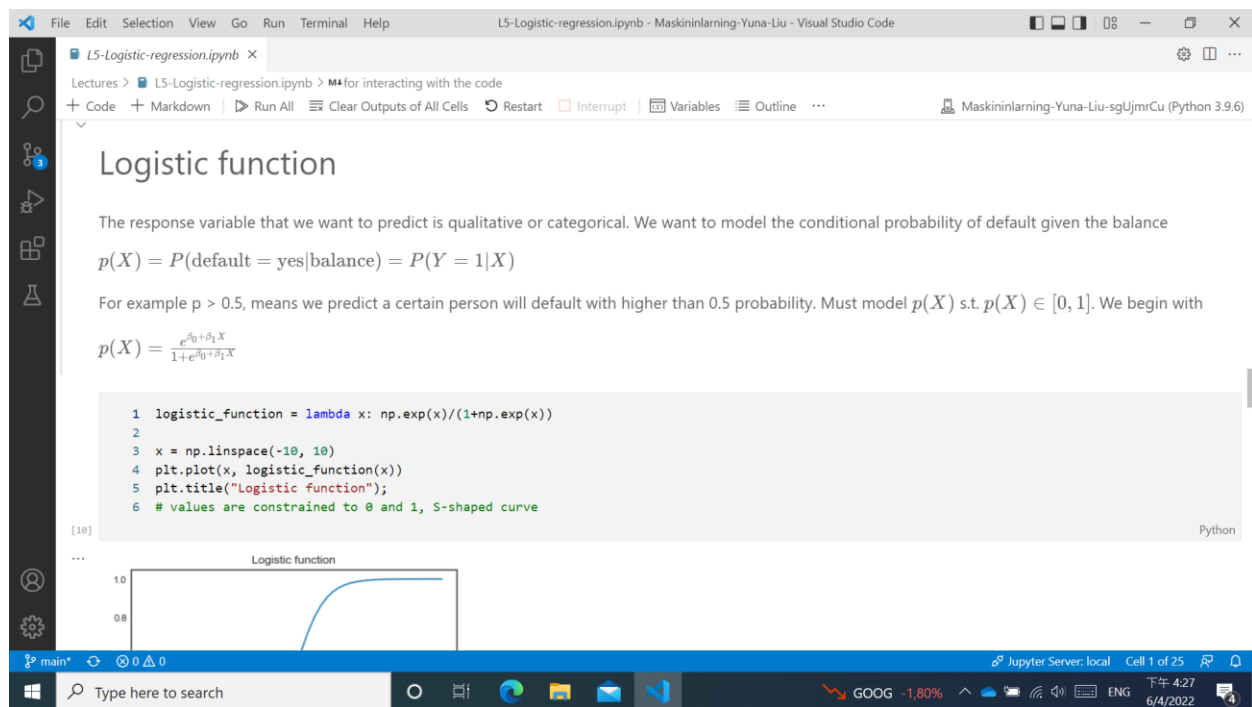
- In general for independent observations Z_1, \dots, Z_n with variance σ^2 each. Mean of the observations is $\bar{Z} = \frac{\sigma^2}{n}$. So by averaging set of observation we reduce the variance.
- we do this by bagging - bootstrap aggregation. We sample multiple times from a training set and average over all the predictions.
- For classification trees we use the majority vote among all the predictions by the bootstrapped trees

Random forest

- grow many decision trees on bootstrapped samples
- randomly choose $m \approx \sqrt{p}$ predictors/features as split candidates
- choose one of these m features for the split
- new m features are chosen for each split

18. Ge ett exempel på ett maskininlärningsproblem där man kan applicera logistisk regression.

For example when we judge a kind of cancer, the response variable will be 0/1. The features can be the size of tumor, number of nearby lymph nodes, metastasis and so on. When $y > 0.5$ we defined positive, otherwise negative.



The screenshot shows a Jupyter Notebook interface with the following content:

Logistic function

The response variable that we want to predict is qualitative or categorical. We want to model the conditional probability of default given the balance

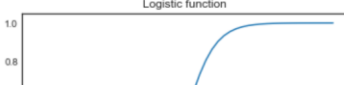
$$p(X) = P(\text{default} = \text{yes} | \text{balance}) = P(Y = 1 | X)$$

For example $p > 0.5$, means we predict a certain person will default with higher than 0.5 probability. Must model $p(X)$ s.t. $p(X) \in [0, 1]$. We begin with

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$$

```
1 logistic_function = lambda x: np.exp(x)/(1+np.exp(x))
2
3 x = np.linspace(-10, 10)
4 plt.plot(x, logistic_function(x))
5 plt.title("Logistic function");
6 # values are constrained to 0 and 1, S-shaped curve
```

[10]



The plot shows the logistic function, which is an S-shaped curve. The x-axis ranges from -10 to 10, and the y-axis ranges from 0 to 1. The curve starts near 0 for negative x and approaches 1 for positive x.

Odds

A convenient way to talk about how a variable relates to another one is to use odds ratio, which is a statistics that can vary from 0 to ∞ . The odds of a default vs not a default is given by this fraction of probability:

$$\frac{p(X)}{1 - p(X)} = e^{\beta_0 + \beta_1 X}$$

For example on average 3 of 10 gets a default is given by $0.3/0.7 = 3/7$.

We take logarithm of both sides and obtain the log odds or logit

$$\log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X$$

This model is linear in X and the parameters β_0, β_1 are chosen to maximize the likelihood function

$$l(\beta_0, \beta_1) = \prod_{y_i=1} p(x_i) \prod_{y_j=0} (1 - p(x_j))$$

In practice the log likelihood function is maximized using an optimization algorithm such as batch gradient descent, mini-batch gradient descent or stochastic gradient descent.

19. Beskriv kort skillnader mellan supervised learning och unsupervised learning.

The main distinction between the two approaches is the use of labeled datasets. To put it simply, supervised learning uses labeled input and output data, while an unsupervised learning algorithm does not.

20. Beskriv kort skillnader mellan regression och klassificering.

Classification is the task of predicting a discrete class label. Regression is the task of predicting a continuous quantity.