# Introduction to Data Analysis and Machine Learning in Physics: 3. Machine Learning Basics, Multivariate Analysis

Jörg Marks

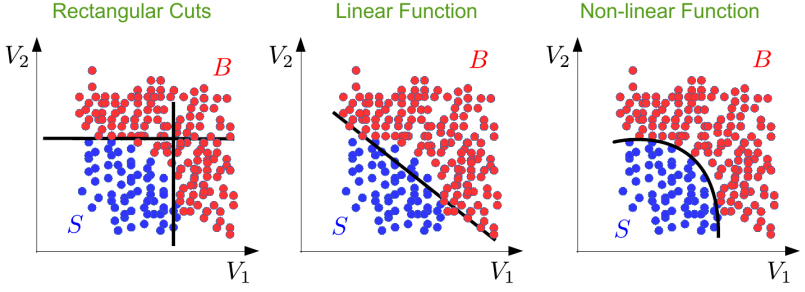
Studierendentage, 7-11 April 2025

# Multi-variate analyses (MVA)

- General Question

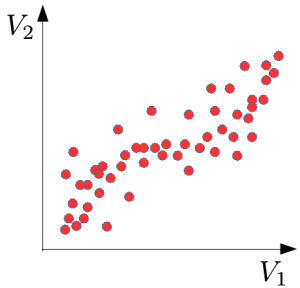
  There are 2 categories of distinguishable data, S and B, described by discrete variables. What are criteria for a separation of both samples?
  - Single criteria are not sufficient to distinguish S and B
  - Reduction of the variable space to probabilities for S or B

- Classification of measurements using a set of observables $(V_1, V_2, ...., V_n)$
  - find optimal separation conditions considering correlations



# Multi-variate analyses (MVA)

- Regression - in the multidimensional observable space $(V_1, V_2, ...., V_n)$ a functional connection with optimal parameters is determined




constant function

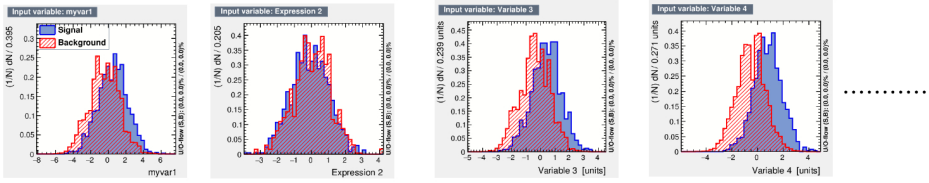
linear function

non-linear function
piecewise definied function
splines

- supervised regression: model is known
- unsupervised regression: model is unknown
- for the parameter determination Maximum likelihood fits are used

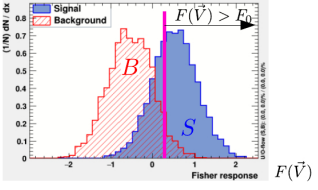
# MVA Classification in N Dimensions

For each event there are N measured variables




$$\vec{V} = \{V_1, V_2, .....V_N\}$$

- Search for a mathematical transformation F of the N dimensional input space to a one dimensional output space $F(\boldsymbol{V}) : \mathbb{R}^N \rightarrow \mathbb{R}$
- A simple cut in F implements a complex cut in the N dimensional variable space
- Determine $F(\boldsymbol{V})$ using a model and fit the parameters

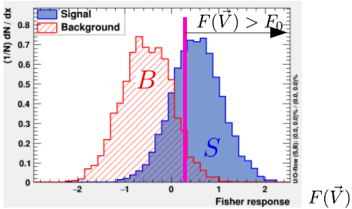


# MVA Classification in N Dimensions

- Parameters
  Important measures to quantify quality

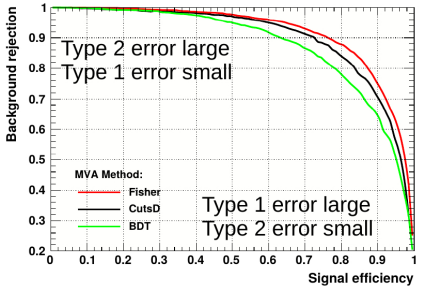
  Efficiency: $\epsilon = \frac{N_S(F>F_0)}{N_s}$

  Purity: $\pi = \frac{N_S(F>F_0)}{(N_s+N_B)(F>F_0)}$



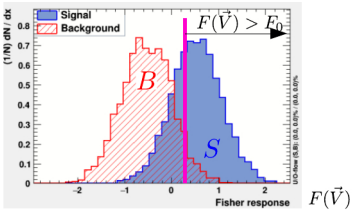

- Reciever Operations Characteristics (ROC)
  Errors in classification

| found cat. / true cat. | detected | Signal | Background |
|---|---|---|---|
| true | Signal | Ok | Type 2 error — loss purity |
| | Background | Type 1 error — loss efficiency | Ok |



# MVA Classification in N Dimensions

- Interpretation of $F(\boldsymbol{V})$
  - The distributions of $F(\boldsymbol{V}|S)$ and $F(\boldsymbol{V}|S)$ are interpreted as probability density functions (PDF), $PDF_S(F)$ and $PDF_B(F)$
  - For a given $F_0$ the probability for signal and background for a given $S/B$ can be determined $P(data = S|F) = \frac{f_S \cdot PDF_S(F)}{f_B \cdot PDF_B(F) + f_S \cdot PDF_S(F)}$

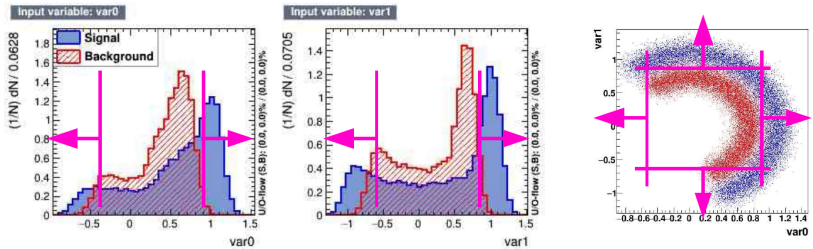



- A cut in the one dimensional Variable $F(\boldsymbol{V}) = F_0$ and accepting all events on the right determines the signal and background efficiency (background rejection). A systematic change of $F(\boldsymbol{V})$ gives the ROC curve.

- A cut in $F(\boldsymbol{V})$ corresponds to a complex hyperplane, which can not neccessarily be described by a function.

# Simple Cuts in Variables

- The most simple classificator to select signal events are cuts in all variables which show a separation
  - The output is binary and not a probability on $S$ or $B$.
  - An optimization of the cuts is done by maximizing of the background suppression for given signal efficiencies.
  - Significance $sig = \epsilon_S \cdot N_S / \sqrt{\epsilon_S \cdot N_S + \epsilon_B(\epsilon_S) N_B}$

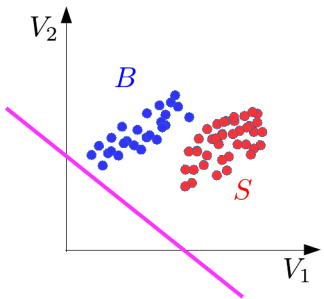


# Fisher Discriminat

Idea: Find a plane, that the projection of the data on the plane gives an optimal separation of signal and background

- The Fisher discriminat is the linear combination of all input variables
  $F(\boldsymbol{V}) = \sum_i w_i \cdot V_i = \boldsymbol{w}^T \boldsymbol{V}$

- $\boldsymbol{w}$ defines the orientation of the plane. The coefficients are defined such that the difference of the expectation values of both classes is large and the variance is small.
  $J(\boldsymbol{w}) = \frac{(F_S - F_B)^2}{\sigma_S^2 + \sigma_B^2} = \frac{\boldsymbol{w}^T K \boldsymbol{w}}{\boldsymbol{w}^T L \boldsymbol{w}}$
  with $K$ as covariance of the the expectation values $F_S - F_B$ and L is the sum
- For the separation a value $F_c$ is determined.



# k-Nearest Neighbor Method (1)

$k$-NN classifier:

- Estimates probability density around the input vector
- $p(\boldsymbol{x}|S)$ and $p(\boldsymbol{x}|B)$ are approximated by the number of signal and background events in the training sample that lie in a small volume around the point $\boldsymbol{x}$

Algorithms finds $k$ nearest neighbors:

$$k = k_s + k_b$$

Probability for the event to be of signal type:

$$p_s(\boldsymbol{x}) = \frac{k_s(\boldsymbol{x})}{k_s(\boldsymbol{x}) + k_b(\boldsymbol{x})}$$

# k-Nearest Neighbor Method (2)

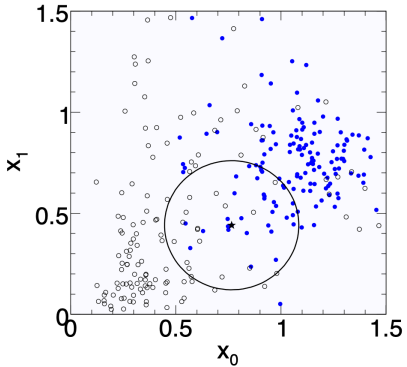
Simplest choice for distance measure in feature space is the Euclidean distance:

$$R = |\boldsymbol{x} - \boldsymbol{y}|$$

Better: take correlations between variables into account:

$$R = \sqrt{(\boldsymbol{x} - \boldsymbol{y})^T V^{-1} (\boldsymbol{x} - \boldsymbol{y})}$$

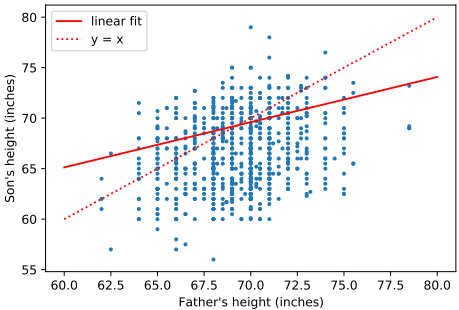
$V =$ covariance matrix, $R =$ "Mahalanobis distance"




The $k$-NN classifier has best performance when the boundary that separates signal and background events has irregular features that cannot be easily approximated by parametric learning methods.

# Linear regression revisited

"Galton family heights data":
origin of the term "regression"




- data: $\{x_i, y_i\}$
- objective: predict $y = f(x)$
- model: $f(x; \boldsymbol{\theta}) = mx + b, \quad \boldsymbol{\theta} = (m, b)$
- loss function:
  $J(\theta|x, y) = \frac{1}{N} \sum_{i=1}^{N} (y_i - f(x_i))^2$
- model training: optimal parameters
  $\hat{\boldsymbol{\theta}} = \arg\min J(\boldsymbol{\theta})$

# Linear regression

- Data: vectors with $p$ components ("features"): $\boldsymbol{x} = (x_1, ..., x_p)$
- $n$ observations: $\{\boldsymbol{x}_i, y_i\}, \quad i = 1, ..., n$
- Prediction for given vector $x$:

$$y = w_0 + w_1 x_1 + w_2 x_2 + ... + w_p x_p \equiv \boldsymbol{w}^\mathsf{T} \boldsymbol{x} \quad \text{where } x_0 := 1$$

- Find weights that minimze loss function:

$$\hat{\boldsymbol{w}} = \min_{\boldsymbol{w}} \sum_{i=1}^{n} (\boldsymbol{w}^\mathsf{T} \boldsymbol{x}_i - y_i)^2$$

- In case of linear regression closed-form solution exists:

$$\hat{\boldsymbol{w}} = (X^\mathsf{T} X)^{-1} X^\mathsf{T} \boldsymbol{y} \quad \text{where } X \in \mathbb{R}^{n \times p}$$

- $X$ is called the design matrix, row $i$ of $X$ is $\boldsymbol{x}_i$

# Linear regression with regularization

- Standard loss function

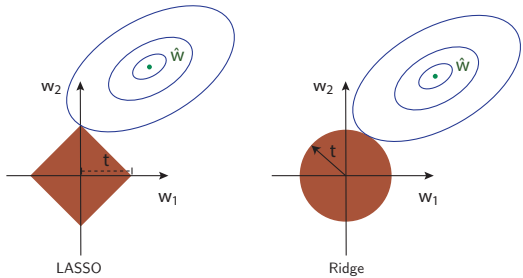
$$C(\boldsymbol{w}) = \sum_{i=1}^{n} (\boldsymbol{w}^\mathsf{T} \boldsymbol{x}_i - y_i)^2$$

- Ridge regression

$$C(\boldsymbol{w}) = \sum_{i=1}^{n} (\boldsymbol{w}^\mathsf{T} \boldsymbol{x}_i - y_i)^2 + \lambda |\boldsymbol{w}|^2$$

- LASSO regression

$$C(\boldsymbol{w}) = \sum_{i=1}^{n} (\boldsymbol{w}^\mathsf{T} \boldsymbol{x}_i - y_i)^2 + \lambda |\boldsymbol{w}|$$




LASSO regression tends to give sparse solutions (many components $w_j = 0$). This is why LASSO regression is also called sparse regression.

# Logistic regression (1)

- Consider binary classification task, e.g., $y_i \in \{0, 1\}$
- Objective: Predict probability for outcome $y = 1$ given an observation $\boldsymbol{x}$
- Starting with linear "score"

$$s = w_0 + w_1 x_1 + w_2 x_2 + ... + w_p x_p \equiv \boldsymbol{w}^\mathsf{T} \boldsymbol{x}$$

- Define function that translates $s$ into a quantity that has the properties of a probability

$$\sigma(s) = \frac{1}{1 + e^{-s}}$$

- We would like to determine the optimal weights for a given training data set. They result from the maximum-likelihood principle.

# Logistic regression (2)

- Consider feature vector $\boldsymbol{x}$. For a given set of weights $\boldsymbol{w}$ the model predicts
  - a probability $p(1|\boldsymbol{w}) = \sigma(\boldsymbol{w}^\mathsf{T}\boldsymbol{x})$ for outcome $y = 1$
  - a probabiltiy $p(0|\boldsymbol{w}) = 1 - \sigma(\boldsymbol{w}^\mathsf{T}\boldsymbol{x})$ for outcome $y = 0$
- The probability $p(y_i|\boldsymbol{w})$ defines the likelihood $L_i(\boldsymbol{w}) = p(y_i|\boldsymbol{w})$ (the likelihood is a function of the parameters $\boldsymbol{w}$ and the observations $y_i$ are fixed).
- Likelihood for the full data sample ($n$ observations)

$$L(\boldsymbol{w}) = \prod_{i=1}^{n} L_i(\boldsymbol{w}) = \prod_{i=1}^{n} \sigma(\boldsymbol{w}^\mathsf{T}\boldsymbol{x})^{y_i} \left(1 - \sigma(\boldsymbol{w}^\mathsf{T}\boldsymbol{x})\right)^{1-y_i}$$

- Maximizing the log-likelihood $\ln L(\boldsymbol{w})$ corresponds to minimizing the loss function

$$C(\boldsymbol{w}) = -\ln L(\boldsymbol{w}) = \sum_{i=1}^{n} -y_i \ln \sigma(\boldsymbol{w}^\mathsf{T}\boldsymbol{x}) - (1 - y_i) \ln(1 - \sigma(\boldsymbol{w}^\mathsf{T}\boldsymbol{x}))$$

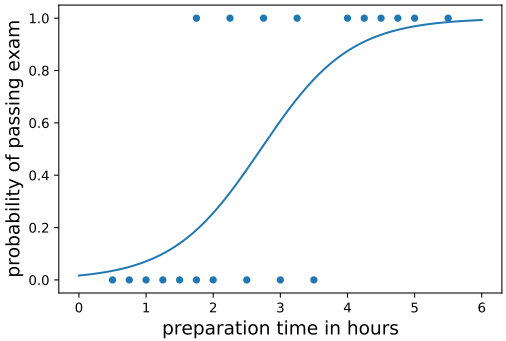
- This is nothing else but the cross-entropy loss function

# Example 1 - Probability of passing an exam (logistic regression) (1)

Objective: predict the probability that someone passes an exam based on the number of hours studying

$$p_{\text{pass}} = \sigma(s) = \frac{1}{1+e^{-s}}, \quad s = w_1 t + w_0, \quad t = \#\text{ hours}$$

- Data set:
  - preparation $t$ time in hours
  - passed / not passes (0/1)
- Parameters need to be determined through numerical minimization
  - $w_0 = -4.0777$
  - $w_1 = 1.5046$




03_ml_basics_logistic_regression.ipynb

# Example 1 - Probability of passing an exam (logistic regression) (2)

Read data from file:

```
# data: 1. hours studies, 2. passed (0/1)
df = pd.read_csv(filename, engine='python', sep='\s+')
x_tmp = df['hours_studied'].values
x = np.reshape(x_tmp, (-1, 1))
y = df['passed'].values
```

Fit the data:

```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression(penalty='l2', fit_intercept=True)
clf.fit(x, y);
```

Calculate predictions:

```
hours_studied_tmp = np.linspace(0., 6., 1000)
hours_studied = np.reshape(hours_studied_tmp, (-1, 1))
y_pred = clf.predict_proba(hours_studied)
```

# Precision and recall

**Precision:**
Fraction of correctly classified instances among all instances that obtain a certain class label.

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

"purity"

**Recall:**
Fraction of positive instances that are correctly classified.

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

"efficiency"

TP: true positives, FP: false positives, FN: false negatives

# Example 2: Heart disease data set (logistic regression) (1)

Read data:

```
filename = "https://www.physi.uni-heidelberg.de/~marks/ml_einfuehrung/Beispiele/heart.csv"
df = pd.read_csv(filename)
df
```

| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 63 | 1 | 3 | 145 | 233 | 1 | 0 | 150 | 0 | 2.3 | 0 | 0 | 1 | 1 |
| **1** | 37 | 1 | 2 | 130 | 250 | 0 | 1 | 187 | 0 | 3.5 | 0 | 0 | 2 | 1 |
| **2** | 41 | 0 | 1 | 130 | 204 | 0 | 0 | 172 | 0 | 1.4 | 2 | 0 | 2 | 1 |
| **3** | 56 | 1 | 1 | 120 | 236 | 0 | 1 | 178 | 0 | 0.8 | 2 | 0 | 2 | 1 |
| **4** | 57 | 0 | 0 | 120 | 354 | 0 | 1 | 163 | 1 | 0.6 | 2 | 0 | 2 | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **298** | 57 | 0 | 0 | 140 | 241 | 0 | 1 | 123 | 1 | 0.2 | 1 | 0 | 3 | 0 |
| **299** | 45 | 1 | 3 | 110 | 264 | 0 | 1 | 132 | 0 | 1.2 | 1 | 0 | 3 | 0 |
| **300** | 68 | 1 | 0 | 144 | 193 | 1 | 1 | 141 | 0 | 3.4 | 1 | 2 | 3 | 0 |
| **301** | 57 | 1 | 0 | 130 | 131 | 0 | 1 | 115 | 1 | 1.2 | 1 | 1 | 3 | 0 |
| **302** | 57 | 0 | 1 | 130 | 236 | 0 | 0 | 174 | 0 | 0.0 | 1 | 1 | 2 | 0 |

303 rows × 14 columns

03_ml_basics_log_regr_heart_disease.ipynb

# Example 2: Heart disease data set (logistic regression) (2)

Define array of labels and feature vectors

```
y = df['target'].values
X = df[[col for col in df.columns if col!="target"]]
```

Generate training and test data sets

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, shuffle=True)
```

Fit the model

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(penalty='l2', fit_intercept=True, max_iter=1000, tol=1E-5)
lr.fit(X_train, y_train)
```

# Example 2: Heart disease data set (logistic regression) (3)

Test predictions on test data set:

```
from sklearn.metrics import classification_report
y_pred_lr = lr.predict(X_test)
print(classification_report(y_test, y_pred_lr))
```

Output:

```
              precision    recall  f1-score   support

           0       0.75      0.86      0.80        63
           1       0.89      0.80      0.84        89

    accuracy                           0.82       152
   macro avg       0.82      0.83      0.82       152
weighted avg       0.83      0.82      0.82       152
```

# Example 2: Heart disease data set (logistic regression) (4)

Compare to another classifier usinf the *receiver operating characteristic* (ROC) curve

Let's take the random forest classifier

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(max_depth=3)
rf.fit(X_train, y_train)
```

Use `roc_curve` from scikit-learn

```
from sklearn.metrics import roc_curve

y_pred_prob_lr = lr.predict_proba(X_test) # predicted probabilities
fpr_lr, tpr_lr, _ = roc_curve(y_test, y_pred_prob_lr[:,1])

y_pred_prob_rf = rf.predict_proba(X_test) # predicted probabilities
fpr_rf, tpr_rf, _ = roc_curve(y_test, y_pred_prob_rf[:,1])
```

# Example 2: Heart disease data set (logistic regression) (5)

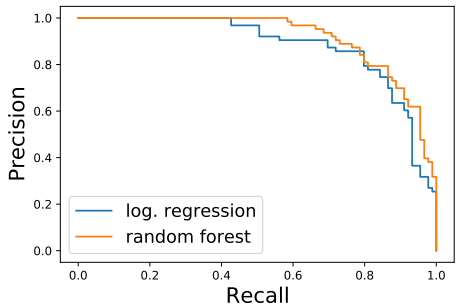
```
plt.plot(tpr_lr, 1-fpr_lr, label="log. regression")
plt.plot(tpr_rf, 1-fpr_rf, label="random forest")
```

Classifiers can be compared with the *area under curve* (AUC) score.

```
from sklearn.metrics import roc_auc_score
auc_lr = roc_auc_score(y_test,y_pred_lr)
auc_rf = roc_auc_score(y_test,y_pred_rf)
print(f"AUC scores: {auc_lr:.2f}, {auc_knn:.2f}")
```

This gives

```
AUC scores: 0.82, 0.83
```



# Multinomial logistic regression: Softmax function

In the previous example we considered two classes (0, 1). For multi-class classification, the logistic function can be generalized to the softmax function.

Now consider $k$ classes and let $s_i$ be the score for class $i$: $\boldsymbol{s} = (s_1, ..., s_k)$

A probability for class $i$ can be predicted with the softmax function:

$$\sigma(\boldsymbol{s})_i = \frac{e^{s_i}}{\sum_{j=1}^{k} e^{s_j}} \quad \text{for} \quad i = 1, ..., k$$

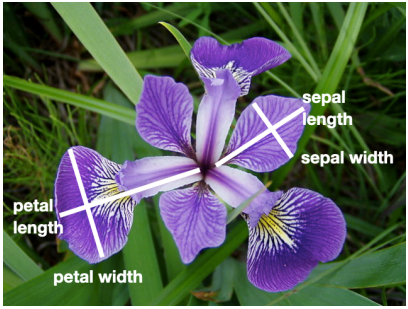
The softmax functions is often used as the last activation function of a neural network in order to predict probabilities in a classification task.

Multinomial logistic regression is also known as softmax regression.

# Example 3: Iris data set (softmax regression) (1)

Iris flower data set

- Introduced 1936 in a paper by Ronald Fisher
- Task: classify flowers
- Three species: iris setosa, iris virginica and iris versicolor
- Four features: petal width and length, sepal width/length, in centimeters




03_ml_basics_iris_softmax_regression.ipynb

https://archive.ics.uci.edu/ml/datasets/Iris

https://en.wikipedia.org/wiki/Iris_flower_data_set

# Example 3: Iris data set (softmax regression) (2)

## Get data set

```
# import some data to play with
# columns: Sepal Length, Sepal Width, Petal Length and Petal Width
iris = datasets.load_iris()
X = iris.data
y = iris.target

# split data into training and test data sets
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=42)
```

## Softmax regression

```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(multi_class='multinomial', penalty='none')
log_reg.fit(x_train, y_train);
```

# Example 3 : Iris data set (softmax regression) (3)

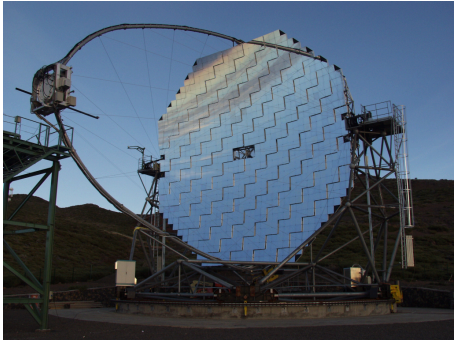
Accuracy and confusion matrix for different classifiers

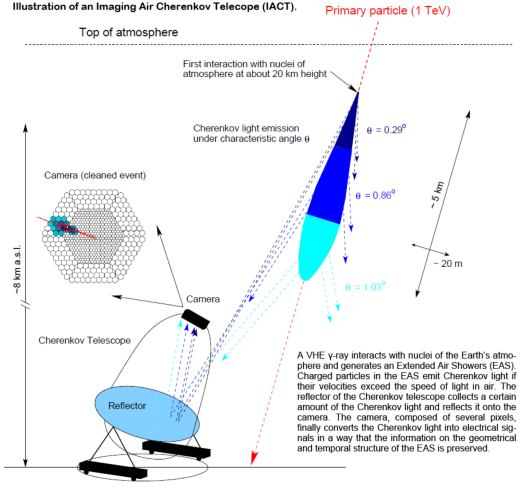
```
for clf in [log_reg, kn_neigh, fisher_ld]:
    y_pred = clf.predict(x_test)
    acc = accuracy_score(y_test, y_pred)
    print(type(clf).__name__)
    print(f"accuracy: {acc:0.2f}")

    # confusion matrix:
    # columns: true class, row: predicted class
    print(confusion_matrix(y_test, y_pred),"\n")
```

```
LogisticRegression
accuracy: 0.96
[[29  0  0]
 [ 0 23  0]
 [ 0  3 20]]

KNeighborsClassifier
accuracy: 0.95
[[29  0  0]
 [ 0 23  0]
 [ 0  4 19]]

LinearDiscriminantAnalysis
accuracy: 0.99
[[29  0  0]
 [ 0 23  0]
 [ 0  1 22]]
```

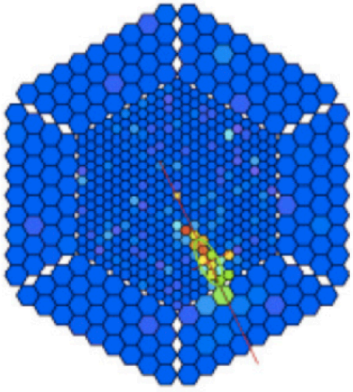
# Exercise 1: Classification of air showers measured with the MAGIC telescope

- Cosmic gamma rays (30 GeV - 30 TeV).
- Cherenkov light from air showers
- Background: air showers caused by hadrons.

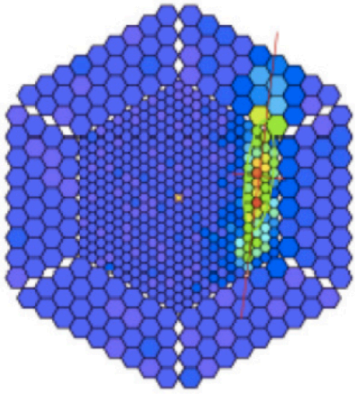




# Exercise 1: Classification of air showers measured with the MAGIC telescope

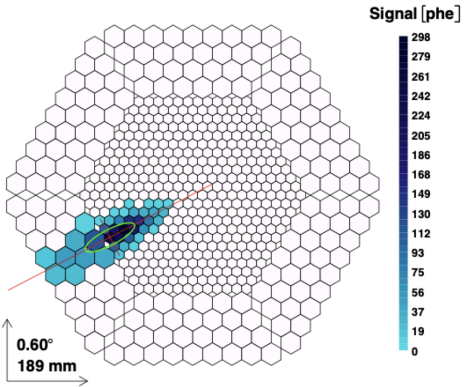


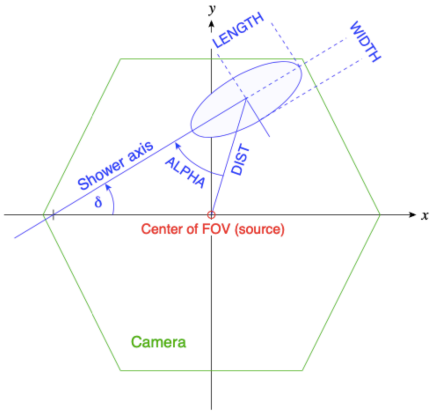
Gamma shower



Hadronic shower

# Exercise 1: Classification of air showers measured with the MAGIC telescope



# Exercise 1: Classification of air showers measured with the MAGIC telescope

## MAGIC data set

https://archive.ics.uci.edu/ml/datasets/magic+gamma+telescope

```
1.  fLength:  continuous  # major axis of ellipse [mm]
2.  fWidth:   continuous  # minor axis of ellipse [mm]
3.  fSize:    continuous  # 10-log of sum of content of all pixels [in #phot]
4.  fConc:    continuous  # ratio of sum of two highest pixels over fSize  [ratio]
5.  fConc1:   continuous  # ratio of highest pixel over fSize  [ratio]
6.  fAsym:    continuous  # distance from highest pixel to center, projected onto major axis [mm]
7.  fM3Long:  continuous  # 3rd root of third moment along major axis  [mm]
8.  fM3Trans: continuous  # 3rd root of third moment along minor axis  [mm]
9.  fAlpha:   continuous  # angle of major axis with vector to origin [deg]
10. fDist:    continuous  # distance from origin to center of ellipse [mm]
11. class:    g,h         # gamma (signal), hadron (background)

g = gamma (signal):     12332
h = hadron (background): 6688

For technical reasons, the number of h events is underestimated.
In the real data, the h class represents the majority of the events.
```

# Exercise 1: Classification of air showers measured with the MAGIC telescope

03_ml_basics_ex_1_magic.ipynb

a) Create for each variable a figure with a plot for gammas and hadrons overlayed.

b) Create training and test data set. The test data should amount to 50% of the total data set.

c) Define the logistic regressor and fit the training data

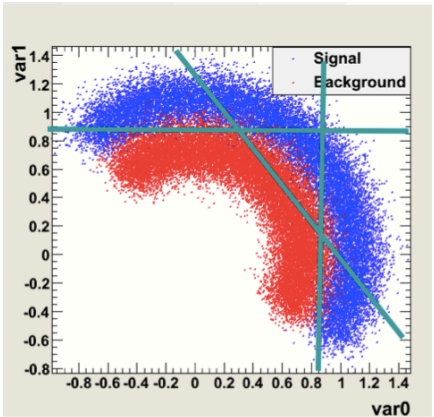
d) Determine the model accuracy and the AUC score

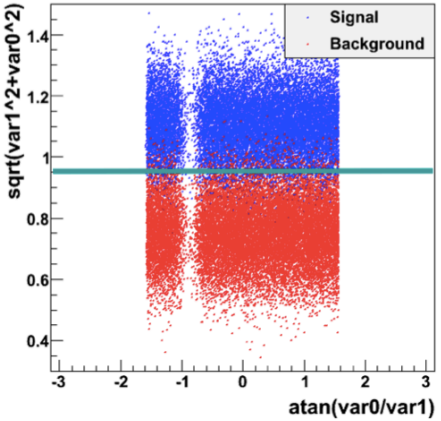
e) Plot the ROC curve (background rejection vs signal efficiency)

# General remarks on multi-variate analyses (MVAs)

- MVA Methods
  - More effective than classic cut-based analyses
  - Take correlations of input variables into account
- Important: find good input variables for MVA methods
  - Good separation power between S and B
  - No strong correlation among variables
  - No correlation with the parameters you try to measure in your signal sample!
- Pre-processing
  - Apply obvious variable transformations and let MVA method do the rest
  - Make use of obvious symmetries: if e.g. a particle production process is symmetric in polar angle $\theta$ use $|\cos\theta|$ and not $\cos\theta$ as input variable
  - It is generally useful to bring all input variables to a similar numerical range

# Example of feature transformation



# Exercise 2: Data preprocessing

a) Read the description of the `sklearn.preprocessing` package.

b) Start from the example notebook on the logistic regression for the heart disease data set (03_ml_basics_log_regr_heart_disease.ipynb). Pre-process the heart disease data set according to the given example. Does preprocessing make a difference in this case?