Machine Intelligence:: Deep Learning
Week 5

Beate Sick, Loran Avci, Oliver Dürr

Institut für Datenanalyse und Prozessdesign

Zürcher Hochschule für Angewandte Wissenschaften

Part II: How to evaluate probabilistic model &
How to model count data

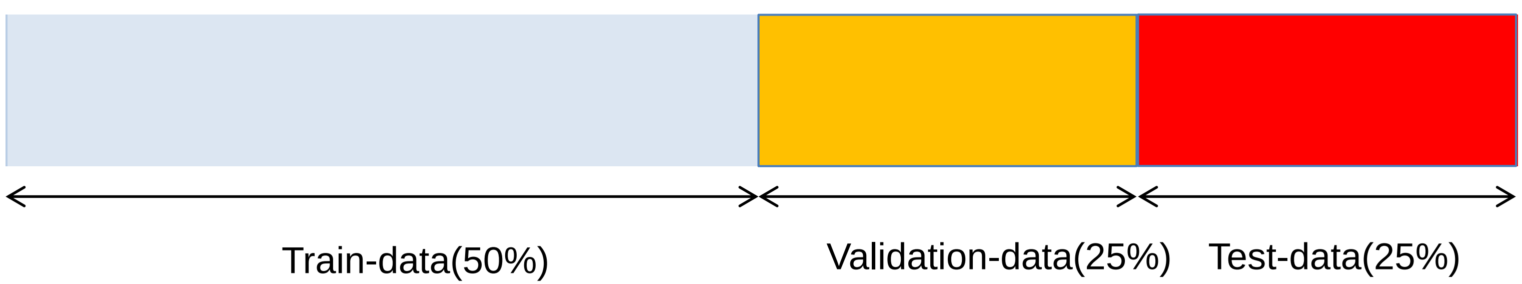
Winterthur, 23. March. 2021

How to evaluate
a probabilistic prediction model?

Check prediction quality on NEW data

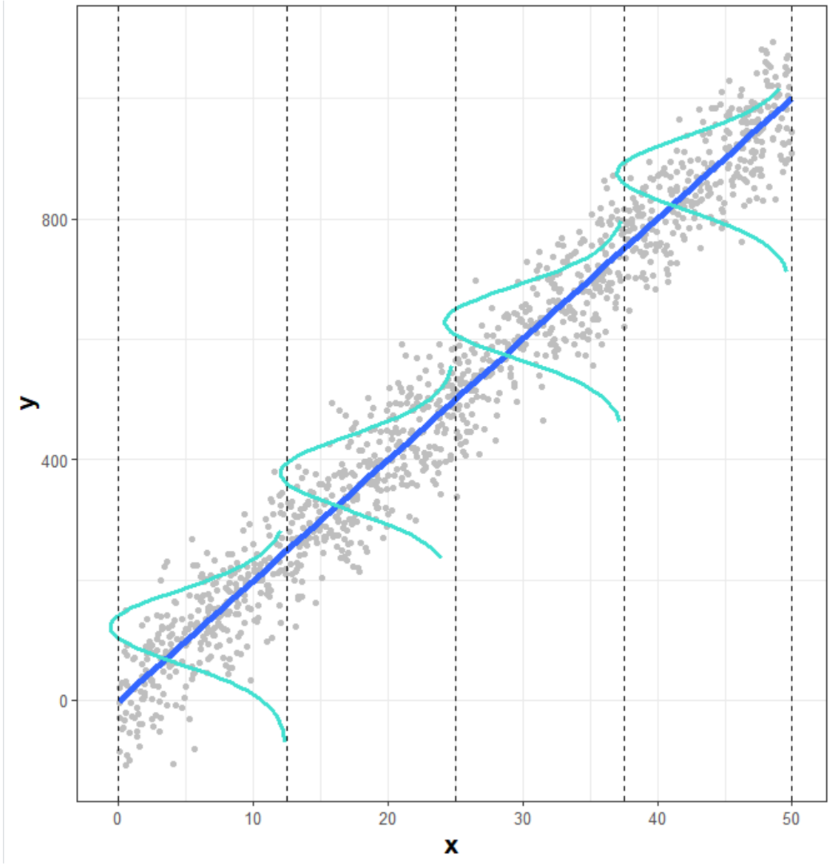
Nils Bohr, physics Nobel price 1922

Common data split:

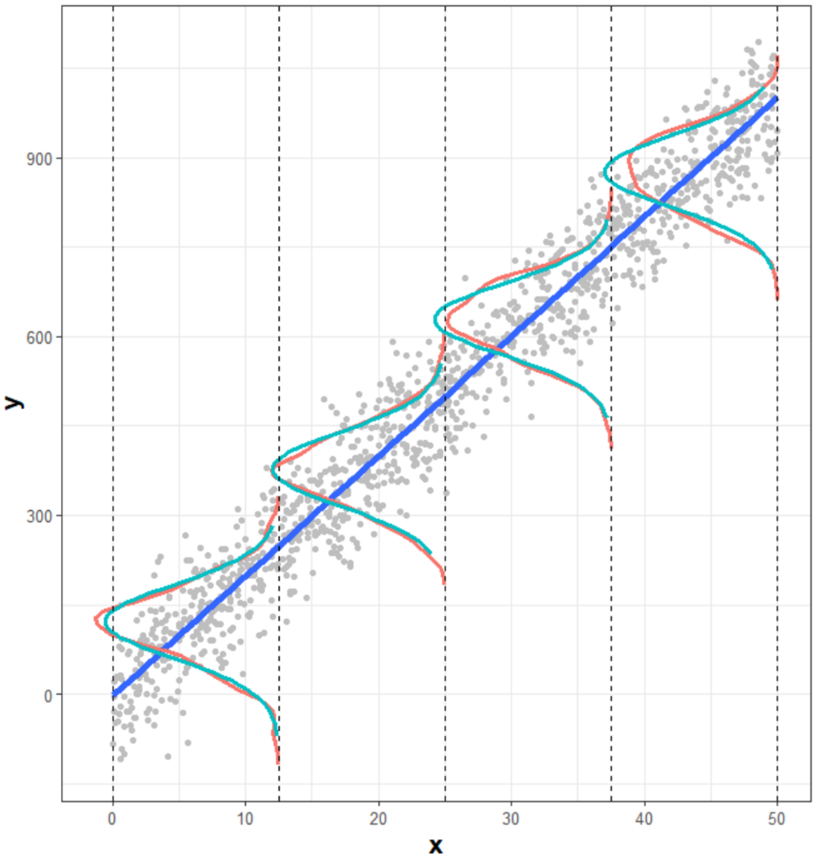


Visually: Do predicted and observed outcome distribution match?

Validation data along with predicted outcome distribution (Gauss with const σ)

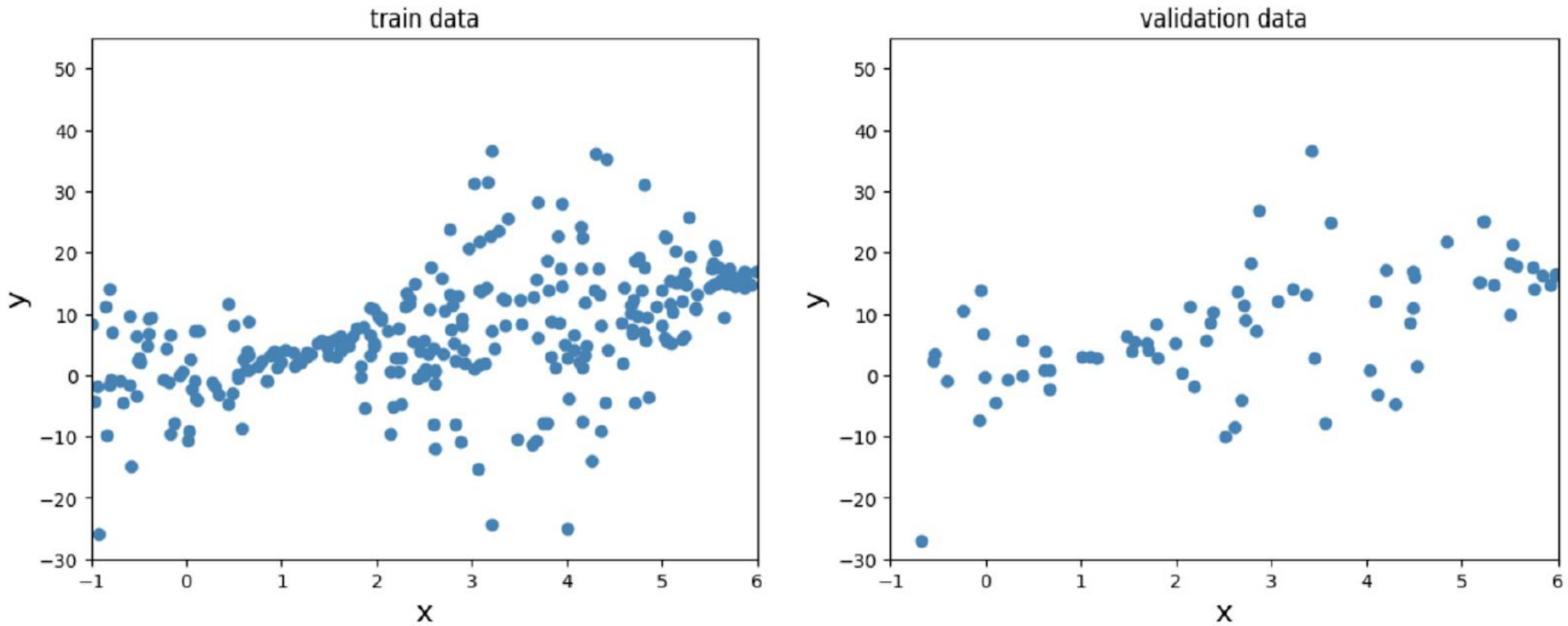


Validation data along with predicted and observed outcome distribution



A large validation data set is needed to ensure underlying assumption:
observed distribution = data generating distribution

Recall simulate data for linear regression models

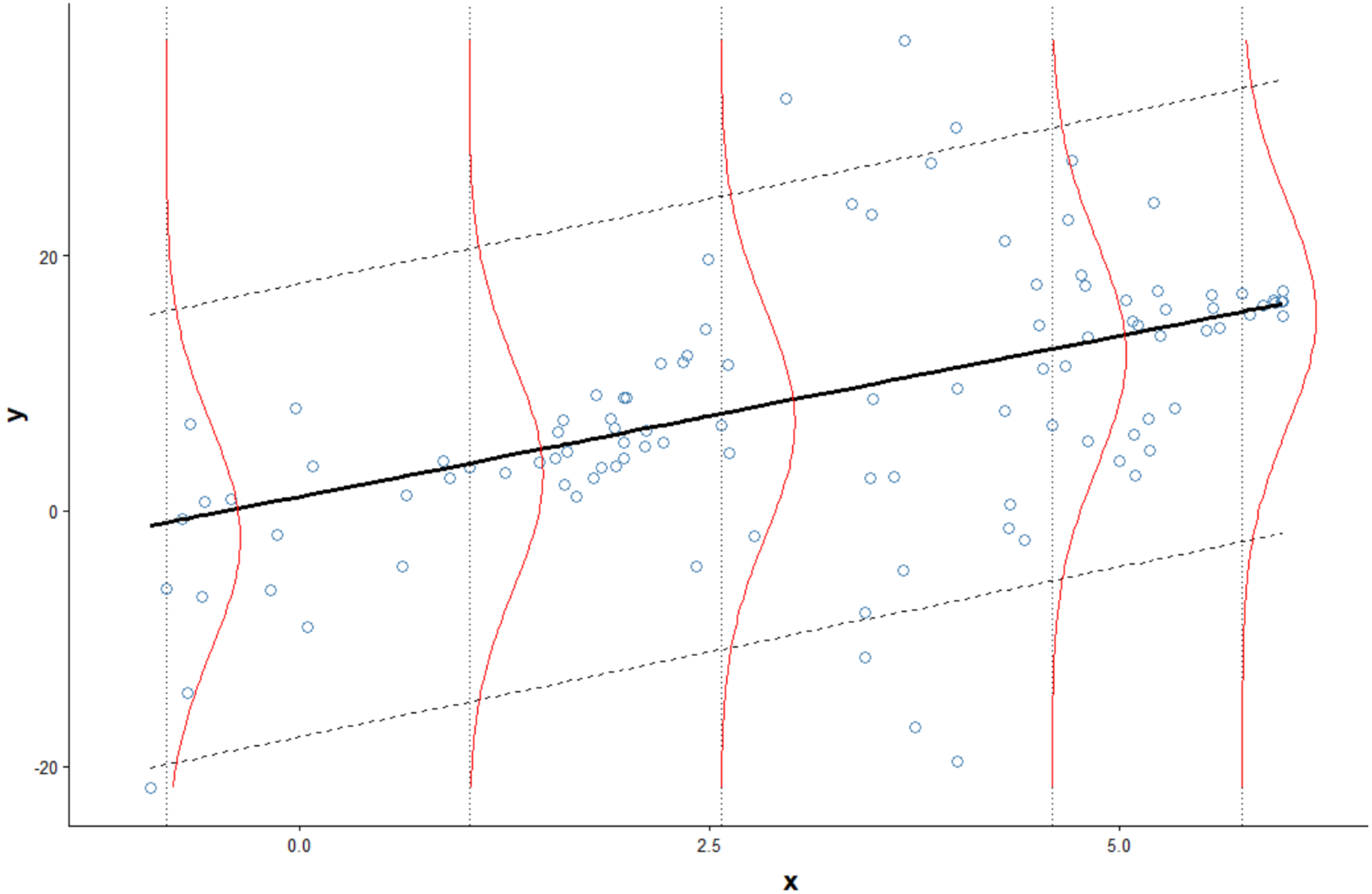


Model_1 (linear regression with constant variance): $(y \mid x) \sim N(\mu_x, \sigma^2)$

Model_2 (linear regression with flexible variance): $(y \mid x) \sim N(\mu_x, \sigma_x^2)$

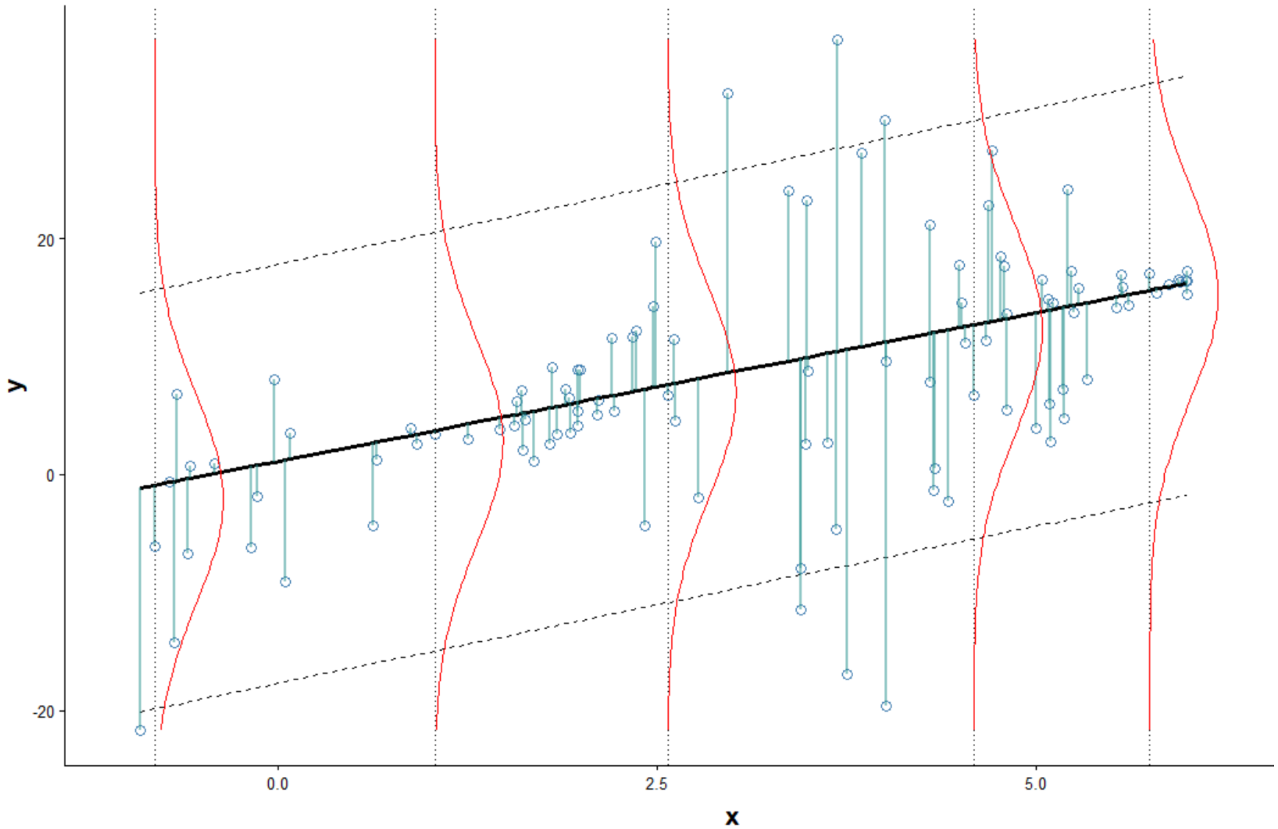
Predicted outcome distribution from model_1 (constant σ)

Validation data



Root mean square error (RMSE) or mean absolute error (MAE)

Validation data



$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{\mu}_{x_i})^2}$$

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{\mu}_{x_i}\right|$$

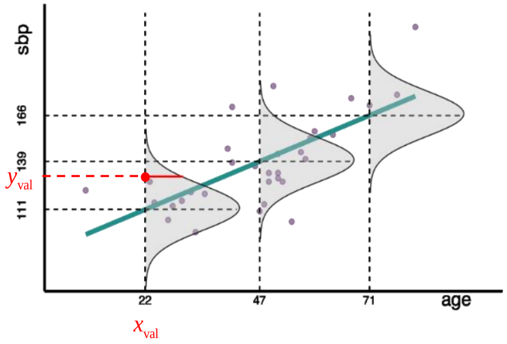
RMSE and MAE alone do not capture performance for probabilistic models!

Both only depend on the mean (μ) of the CPD, but not on it's shape or spread (σ) and are not appropriate to evaluate the quality of the predicted distribution of a probabilistic model.

Use the NLL to score a probabilistic prediction model

A score *S* takes the CPD and *one test instance* and yields a real number
(the smaller the score the better is the predicted CPD)

$$S_{\text{NLL}}\left(p(y\,|\,x_{val}), y_{val}\right) = -\log\left(p(y_{val}\,|\,x_{val})\right)$$



Definition of the term "strictly proper": A strictly proper score is then and only then minimal, when the predicted CPD matches the true CPD.

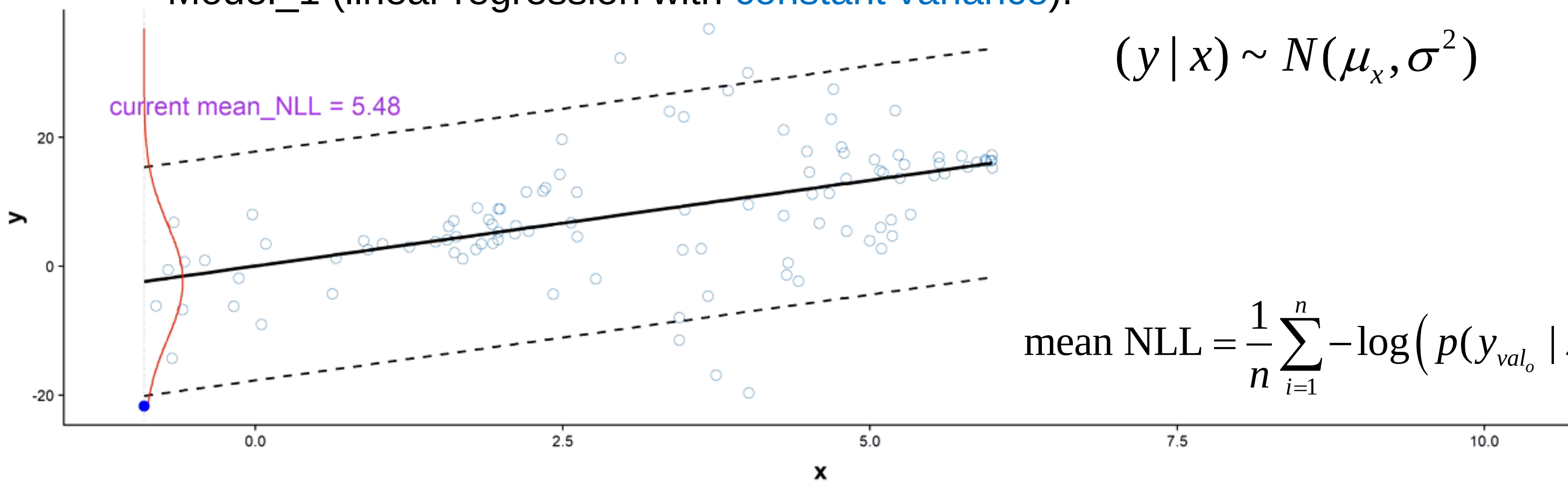
It is provable that the negative log-likelihood (NLL, aka log-score) is the only smooth, proper and local score for continuous variables

(Bernardo, J. M., 1979: Expected information as expected utility. *Ann. Stat.,* **7,** 686–690)

Use validation NLL to compare probabilistic models

Model_1 (linear regression with constant variance):

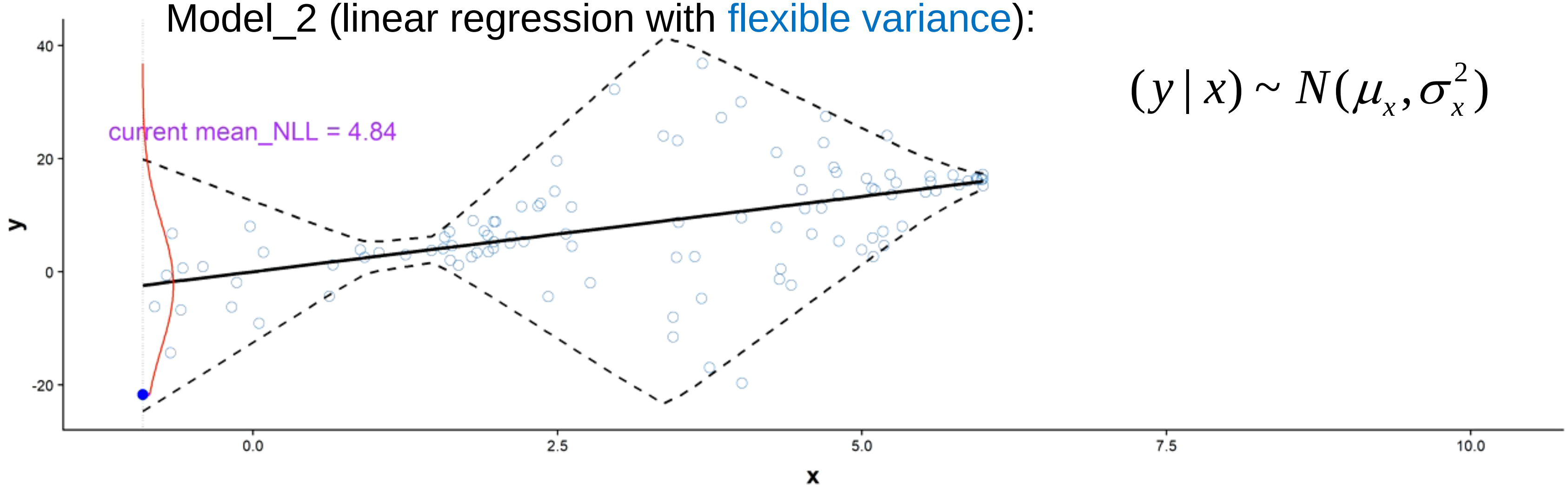
$$(y \mid x) \sim N(\mu_x, \sigma^2)$$



$$\text{mean NLL} = \frac{1}{n}\sum_{i=1}^{n} -\log\Big(p(y_{val_o} \mid x_{val_i})\Big)$$

Model_2 (linear regression with flexible variance):

$$(y \mid x) \sim N(\mu_x, \sigma_x^2)$$



[Tensorchief's youtu.be channel]

Modeling count data

The camper example

N=250 groups visiting a national park

Y=count: number of fishes cought

X1=persons: number of persons in group

X2=child: number of children in the group

X3=bait: indicates of life bait was used

X4=camper: indicates if camper is brought

Data: https://stats.idre.ucla.edu/r/dae/zip

Modeling count data:

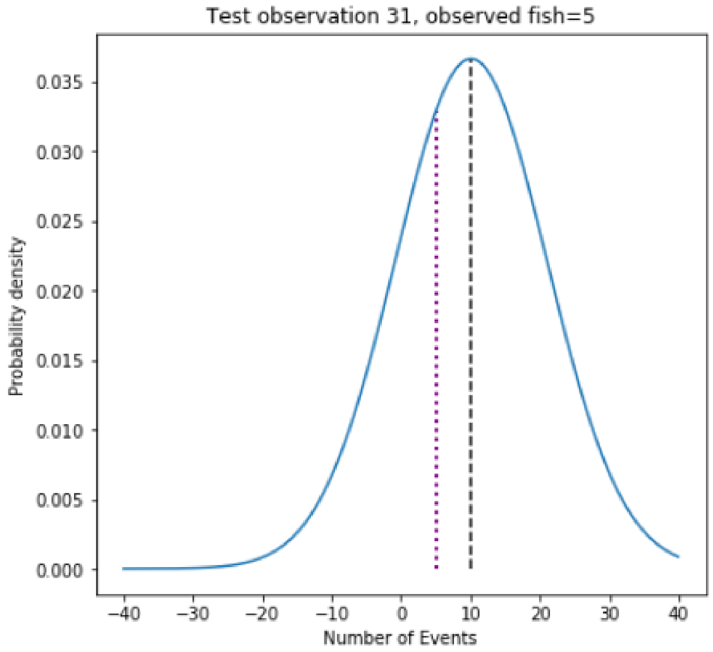
M1: Linear regression

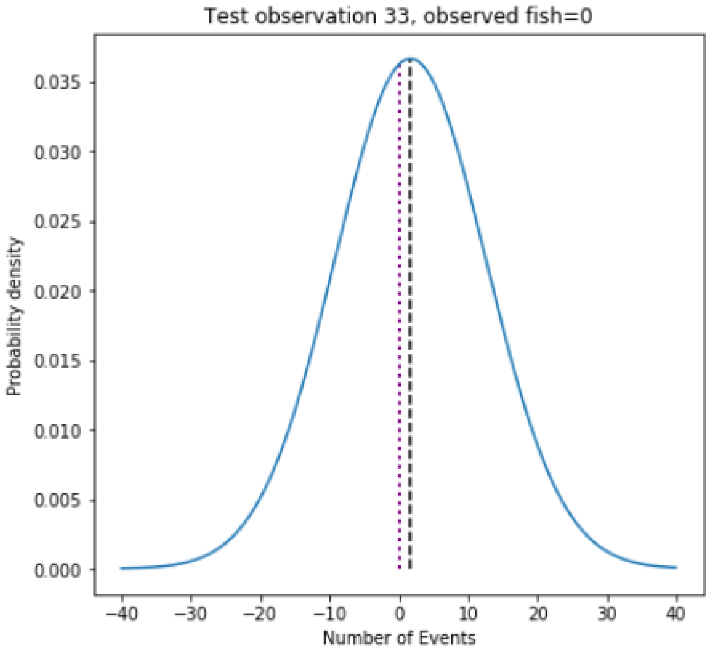
Model 1: linear regression, get test NLL from Gaussian CPD

```
model_lr = Sequential()
model_lr.add(Dense(1,input_dim=d, activation='linear'))
model_lr.compile(loss='mean_squared_error',optimizer=Adam(learning_rate=0.01))
```

Predict CPD for outcome in test data:

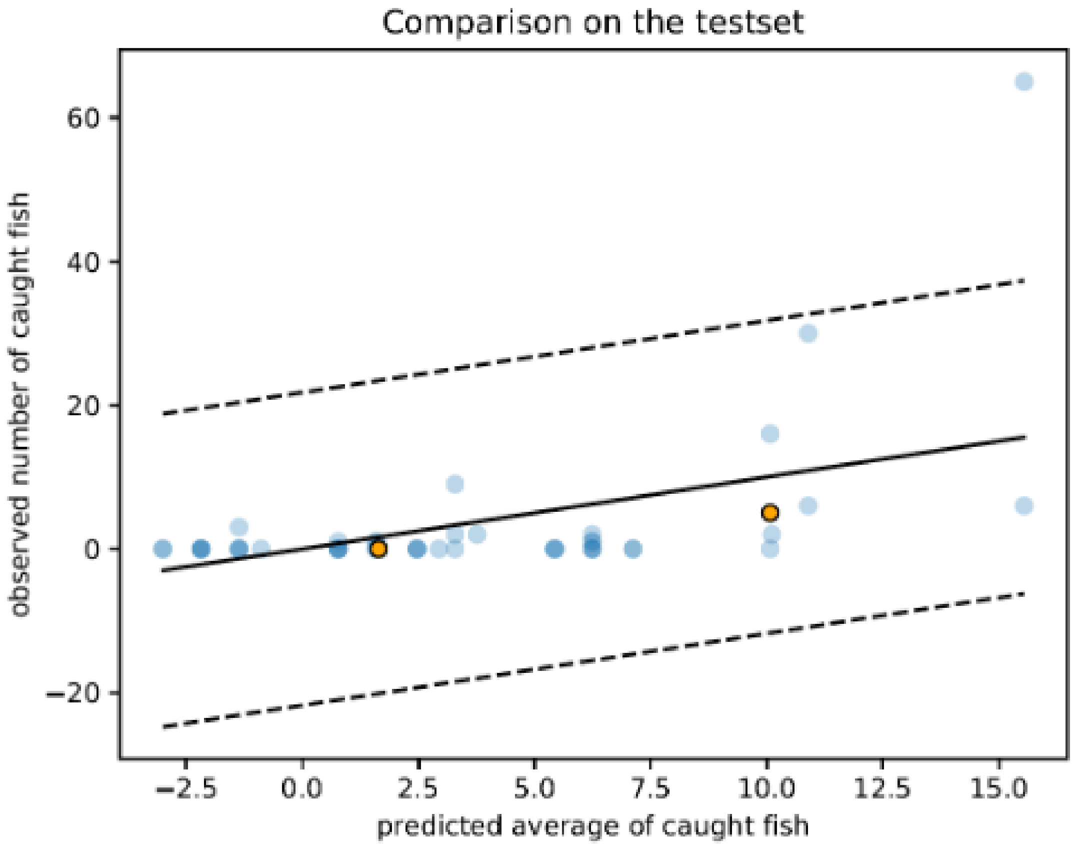
Group 31 used livebait, had a camper and were 4 persons with one child. Y=5 fish.
Group 33 used livebait, didn't have a camper and were 4 persons with two childern. Y=0 fish.





What is the likelihood of the observed outcome in test obs 31 and 33?

Model 1: linear regression, visualize the CPDs by quantiles



How do you read this graph?

Is a Gaussian CPD appropriate to model count data?

A Gaussian CPD is not ideal, because:

- Counts can only take integer values, but a Gaussian assigns likelihoods also to values between two integers

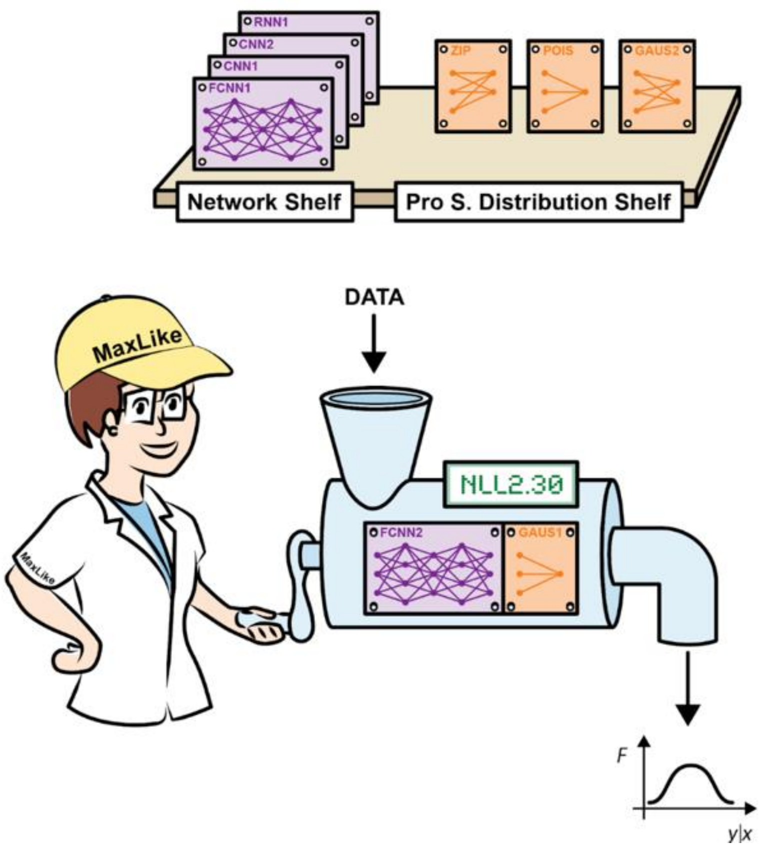
- If counts result from a Poisson process then the mean is equal to the variance of the distribution, which is not taken into account by a Gaussian CPD

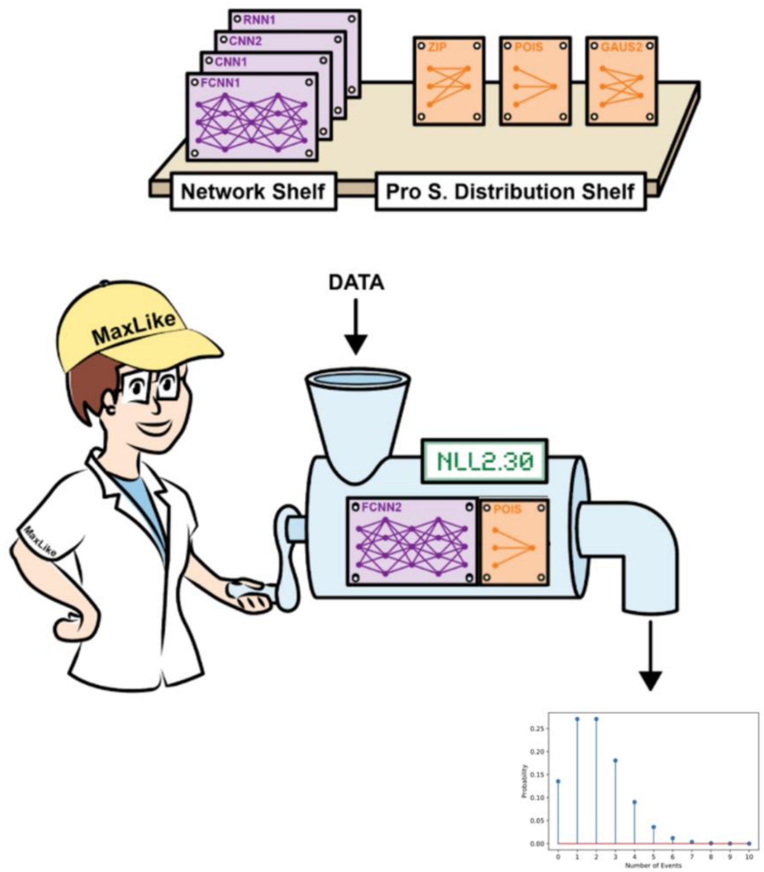
A Gaussian CPD works often quite ok, because:

- For large counts the discrete nature of the counts do not matter so much (looks like normal rounding of continuous values)

- When log-transforming the outcome Y the variance of a Poisonian variable does not anymore depend on the mean and can therefore be modeled by a Gaussian CPD

Probablistic regression: from continuous outcome to counts
→ change the tfp head of your DL model





Modeling count data:

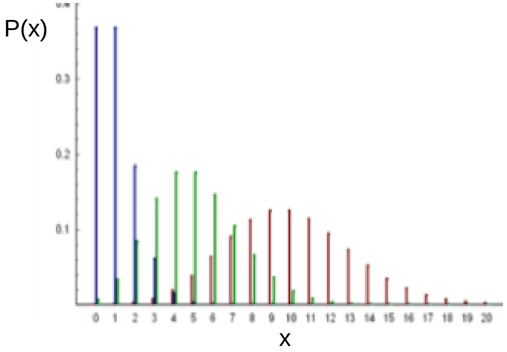
M2: Poisson regression

Recall the classical Poisson model for count data

X: number of incidences **per time unit**

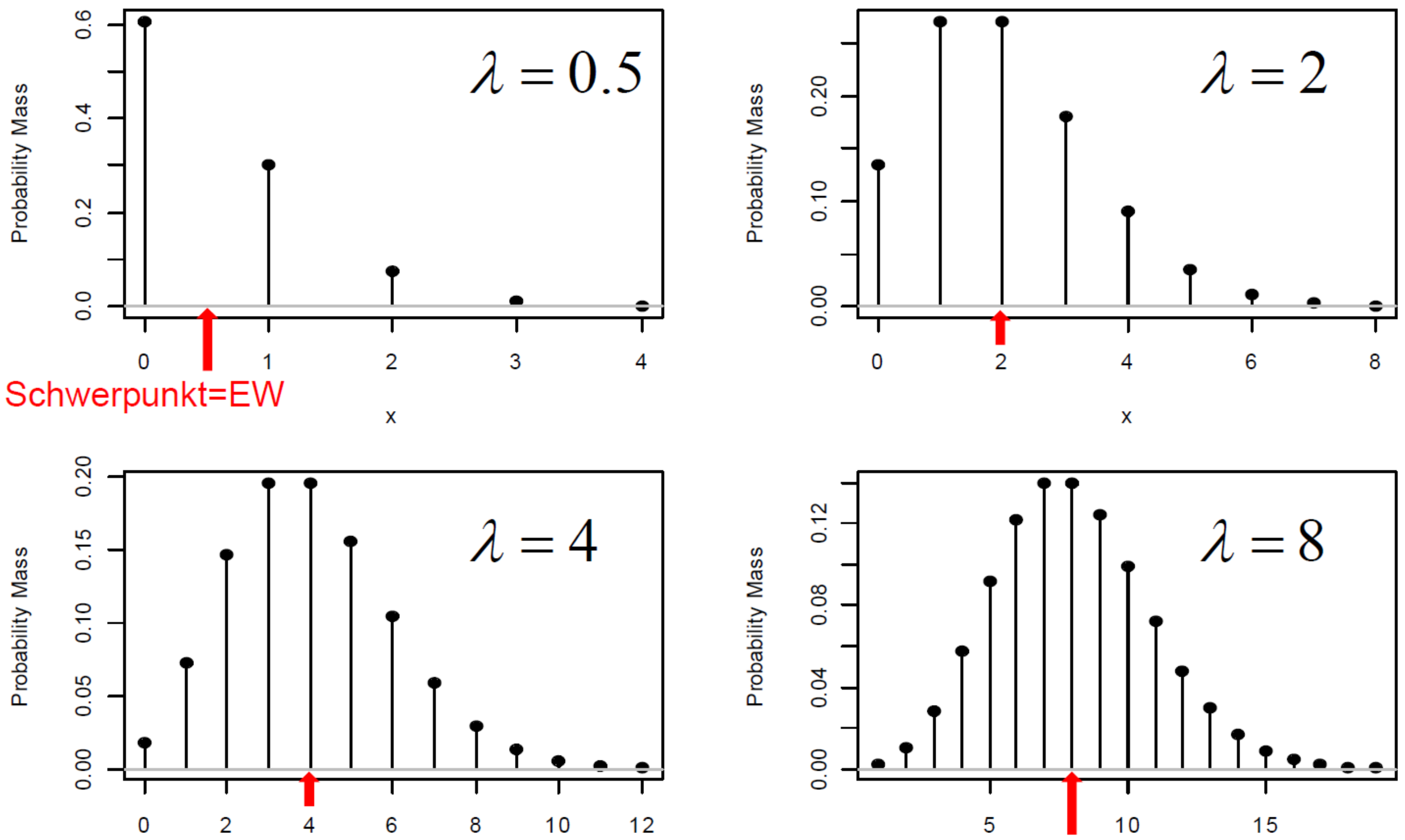
The Poisson model is appropriate to model **counts X per unit**, assuming that
1) the unknown incidence rate λ (per time unit) is constant and
2) the incidences occur independently

$$P(X = x) = \frac{\lambda^x}{x!} e^{-\lambda} \quad , \qquad x = 0, 1, 2, \ldots$$



$$E(X) = Var(X) = \lambda$$: expected number of counts per time unit

The shape and mean of the Poisson distribution depends on λ



The Poisson distribution in tfp

```
dist = tfd.poisson.Poisson(rate = 2) #A

vals = np.linspace(0,10,11) #B

p = dist.prob(vals) #C

print(dist.mean().numpy())  #D

print(dist.stddev().numpy())   #E

#A Poisson distribution with parameter rate = 2
#B Integer values from 0 to 10 for the x-axis in figure 5.13
#C Computes the probability for the values
#D The mean value, yielding 2.0
#E The standard deviation, yielding sqrt(2.0) = 1.41...
```

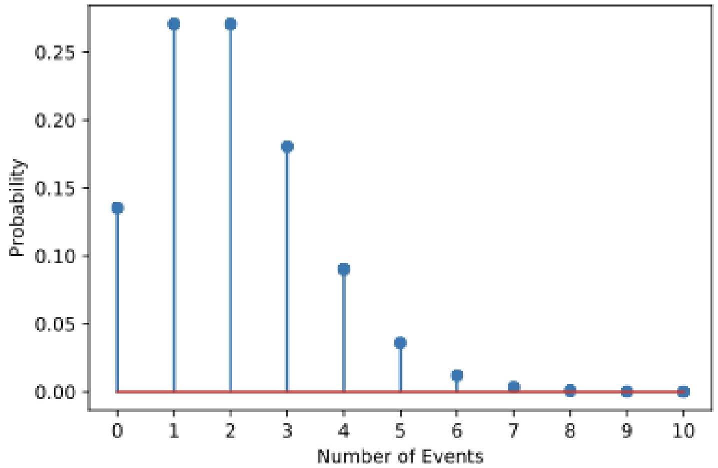


Figure 5.13 Poisson distribution for the case that there are, on average, two events per unit

Poisson regression for count data

Goal: Predict a Poissonian CPD for (Y|X=x) which depends on predictor values

CPD: $Y_{X_i} = (\mathrm{Y}|\mathrm{X}_i) \sim \mathrm{Pois}(\lambda_{x_i})$

We only need to model λ_x to fix the Poissonian CPD!

Model:

$$\log(\lambda_i) = \underbrace{\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}}_{\textbf{Linear predictor } \eta_i}$$

link-function: ensures positive λ after back-transformation

CPD encoded by Poisson regression

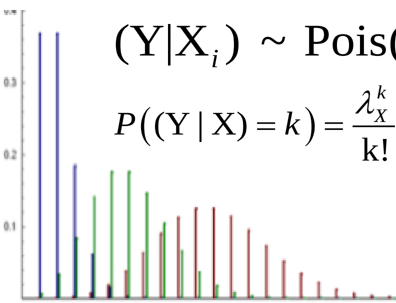
$$\text{CPD:} \quad Y_{X_i} = (Y|X_i) \sim \text{Pois}(\lambda_{x_i})$$

$$Y_x \in \mathbb{N}_0 \quad , \quad \lambda_x \in \mathbb{R}$$

$$\log\left(E(Y_{x_i})\right) = \log(\lambda_{x_i}) = \beta_0 + \beta_1 x_{i1}$$

$$E(Y_{x_i}) = \text{Var}(Y_{x_i}) = \lambda_{x_i} = e^{\beta_0 + \beta_1 x_{i1}}$$

The predicted value of the Poisson regression gives the only one parameter of the CPD: λ_x that depends on the predictor values. We have no error term in the regression formula since the uncertainty of the outcome (counts) is given by the probabilistic Poisson model Pois(λ_x)

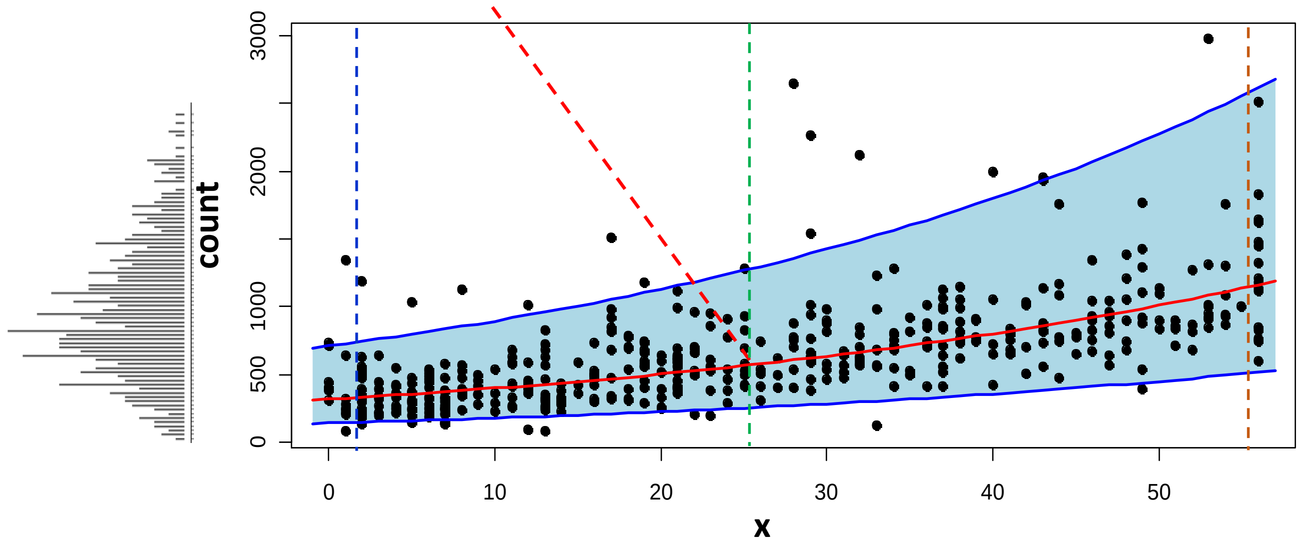


$$(Y|X_i) \sim \text{Pois}(\lambda_{x_i})$$

$$P\left((Y \mid X) = k\right) = \frac{\lambda_X^k}{k!} e^{-\lambda_X} \quad , \quad k = 0, 1, 2, \ldots$$

$$Y \sim V_{arbirary}^{discrete}$$

$$(Y|X_i) \sim \text{Pois}(\lambda_{x_i})$$



Model 2: Poisson regression via NNs in keras

We use a NN without hidden layer to control the rate λ.

```
inputs = Input(shape=(X_train.shape[1],))
rate = Dense(1,
        activation=tf.exp)(inputs) #A
p_y = tfp.layers.DistributionLambda(tfd.Poisson)(rate)

model_p = Model(inputs=inputs, outputs=p_y) #C


def NLL(y_true, y_hat): #D
  return -y_hat.log_prob(y_true)

model_p.compile(Adam(learning_rate=0.01), loss=NLL)
model_p.summary()
```

Using the exp-activation ensures that the rate λ is a positive number

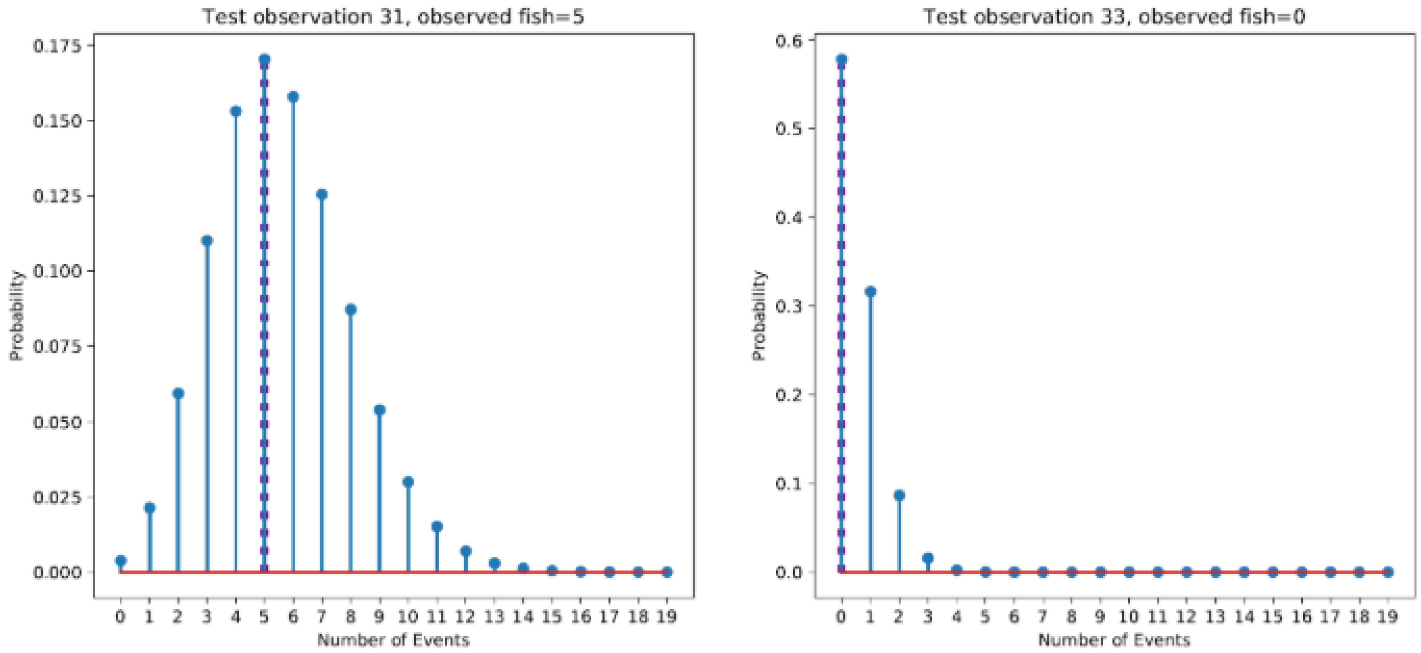
Glueing the NN and the output layer together. Note that output p_y is a tf.distribution

The second argument is the output of the model and thus a TFP distribution. It's as simple as calling log_prob to calculate the log probability of the observation that's needed to calculate the NLL

Model 2: Poisson regression, get test NLL from Gaussian CPD

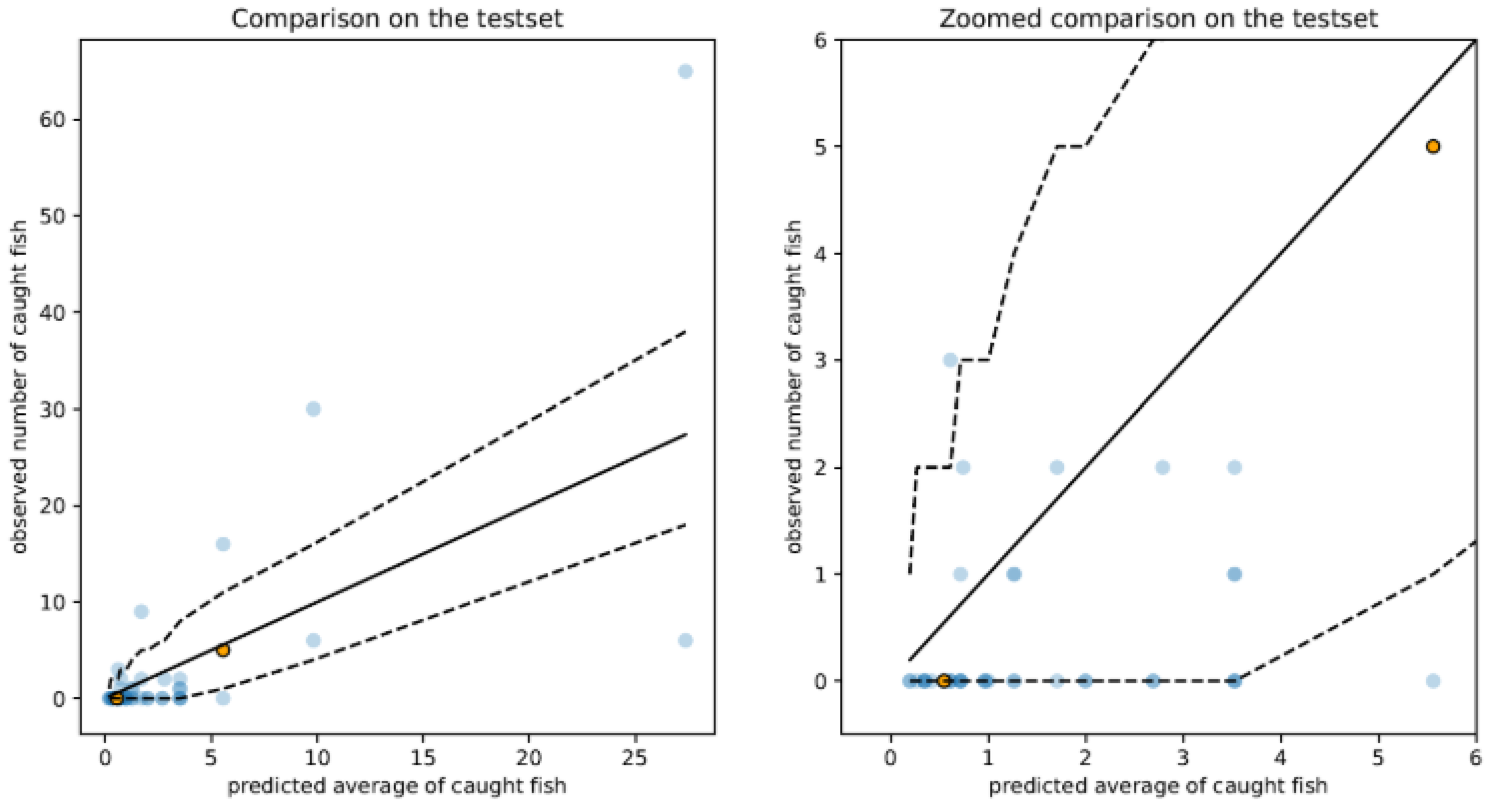
Predict CPD for outcome in test data:

Group 31 used livebait, had a camper and were 4 persons with one child. Y=5 fish.
Group 33 used livebait, didn't have a camper and were 4 persons with two childern. Y=0 fish.



What is the likelihood of the observed outcome in test obs 31 and 33?

Model 2: Poisson regression, visualize the CPDs by quantiles



The mean of the CPD is depicted by the solid lines.
The dashed lines represent the 0.025 and 0.975 quantiles, yielding the borders of a 95% prediction interval.
Note that different combinations of predictor values can yield the same parameters of the CPD.

Approach 2 Model the CPD

- Machen Sie Aufgabe 14_poisreg_with_tfp.ipynb

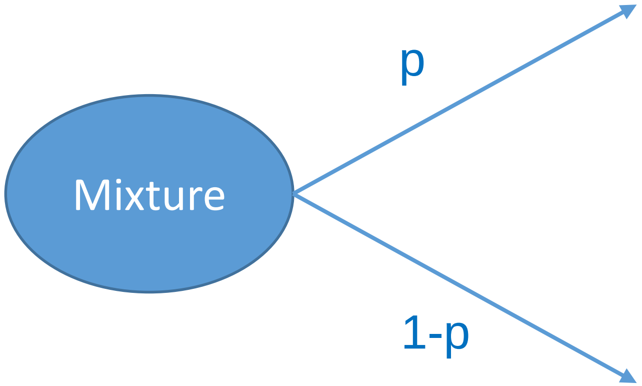
- https://github.com/tensorchiefs/dl_course_2021/blob/master/notebooks/14_poisreg_with_tfp.ipynb

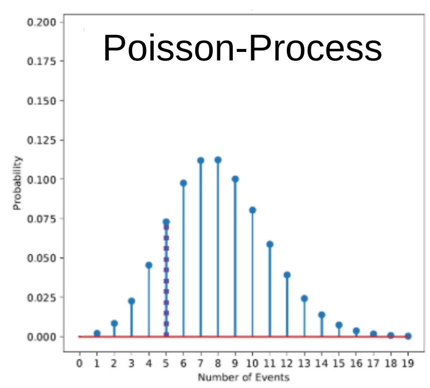
Modeling count data:

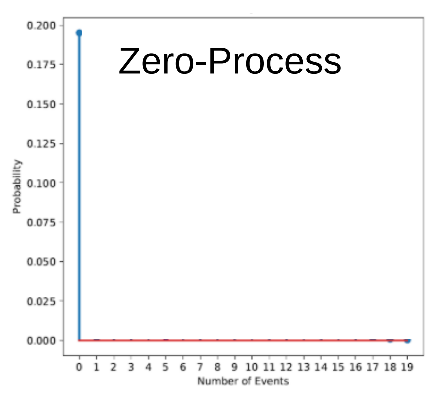
M3: ZIP regression

Zero-Inflated Poisson (ZIP) as Mixture Process

How many fish a group catches does not only depend on luck ;-)

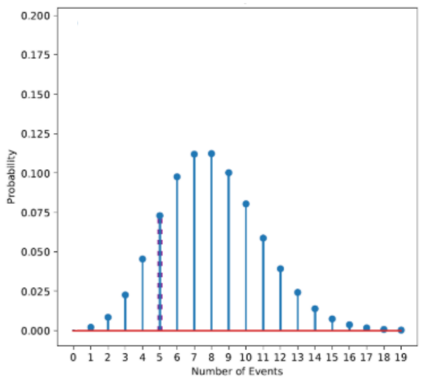




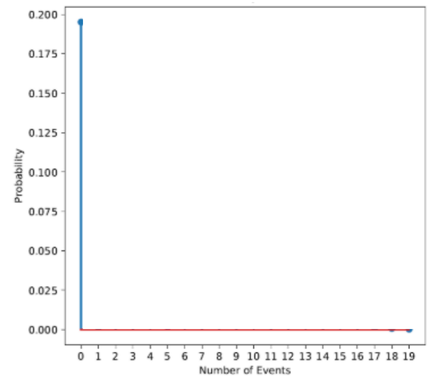


Zero-Inflated Poisson (ZIP) can be seen as Mixture Distribution

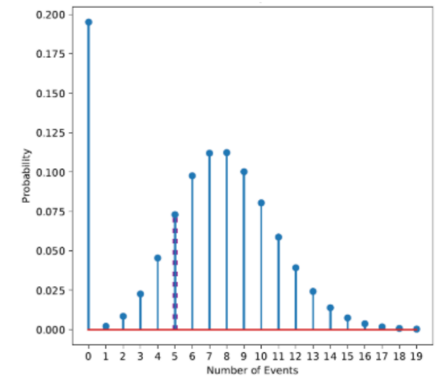
Poisson-Process

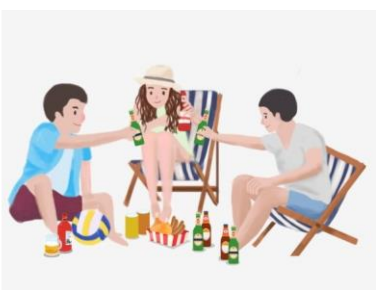
$p \cdot$ 

Zero-Process

$+(1-p) \cdot$ 

Zero-inflated Poisson

$=$ 



Custom distribution for a ZIP distribution

```
def zero_inf(out):
    rate = tf.squeeze(tf.math.exp(out[:,0:1])) # First NN output controls lambda. Exp guarantee value >0
    s = tf.math.sigmoid(out[:,1:2]) # Second NN output controls p; sigmoid guarantees value in [0,1]
    probs = tf.concat([1-s, s], axis=1) # The two probabilities for 0's or Poissonian distribution
    return tfd.Mixture(
          cat=tfd.Categorical(probs=probs),# tfd.Categorical allows creating a mixture of two components
          components=[
          tfd.Deterministic(loc=tf.zeros_like(rate)), # Zero as a deterministic value
          tfd.Poisson(rate=rate), # Value drawn from a Poissonian distribution
        ])
```

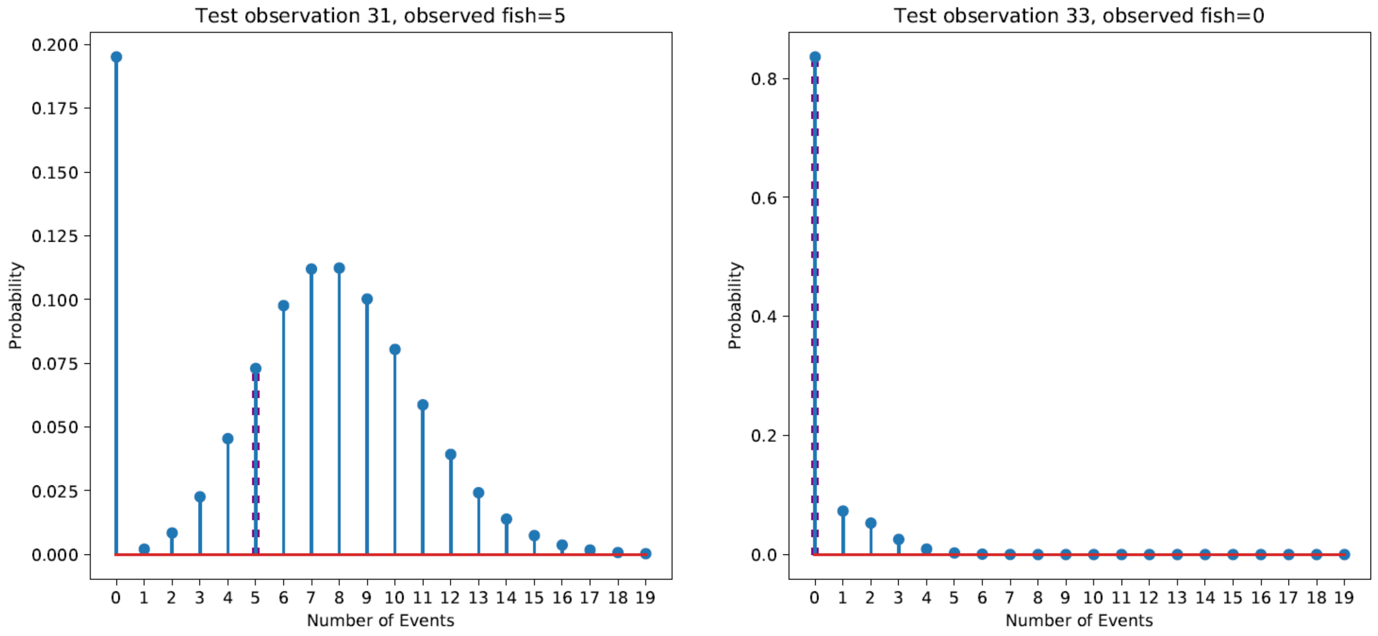
Model 3: Zero-Inflated Poisson regression via NNs in keras

```
## Definition of the custom parameterized distribution

inputs = tf.keras.layers.Input(shape=(X_train.shape[1],))

out = Dense(2)(inputs) #A

p_y_zi = tfp.layers.DistributionLambda(zero_inf)(out)

model_zi = Model(inputs=inputs, outputs=p_y_zi)
```

Model 3: ZIP regression, get test NLL from Gaussian CPD

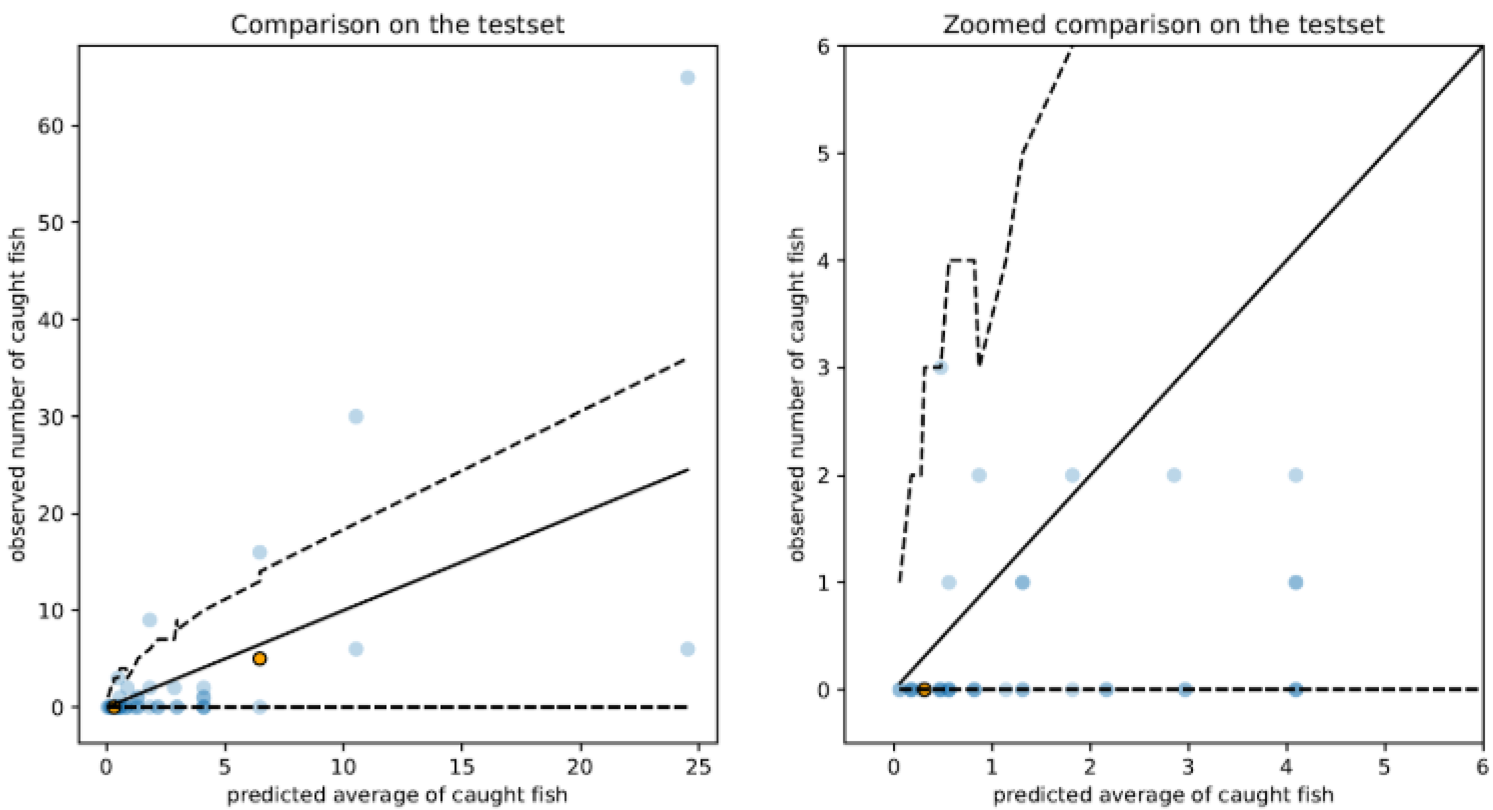
Predict CPD for outcome in test data:

Group 31 used livebait, had a camper and were 4 persons with one child. Y=5 fish.
Group 33 used livebait, didn't have a camper and were 4 persons with two childern. Y=0 fish.



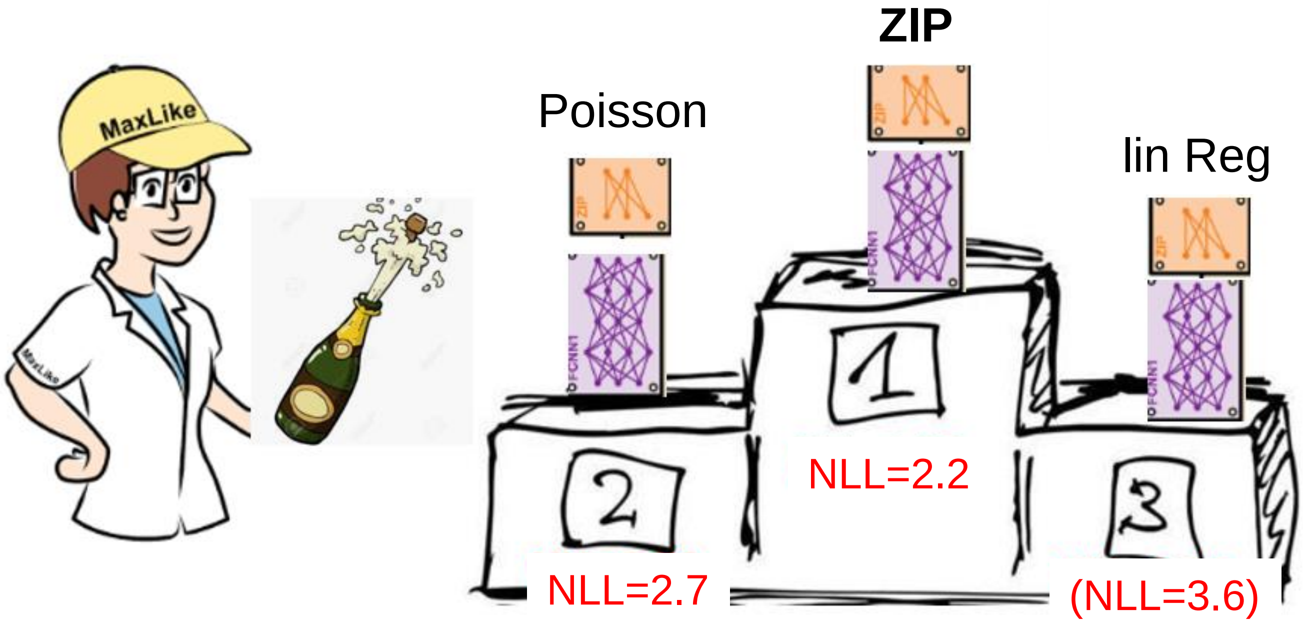
What is the likelihood of the observed outcome in test obs 31 and 33?

Model 3: ZIP regression, visualize the CPDs by quantiles



The mean of the CPD is depicted by the solid lines.
The dashed lines represent the 0.025 and 0.975 quantiles, yielding the borders of a 95% prediction interval.
Note that different combinations of predictor values can yield the same parameters of the CPD.

Validation NLL allows to rank different probabilistic models



Summary

- A probabilistic model predicts for each input a whole conditional probability distribution (CPD).
- The predicted CPD assigns for each possible outcome *y*, a probability with which it's expected.
- The negative log-likelihood (NLL) measures how well the CPD matches the actual distribution of the outcomes.
- The NLL is used as a loss function when training a probabilistic model.
- The NLL on new data is used to measure, and to compare, the prediction performance of different probabilistic models.
- Using a proper choice for the CPD enhances the performance of your models.
- For continuous data, a common first choice is a Normal distribution
- For count data, common choices for distribution are Poisson, Negative-Binomial, or Zero-inflated Poisson (ZIP).