# Tutorial 9

Statistical Computation and Analysis

Spring 2025

# Tutorial Outline

- Model comparison
  - Widely applicable information criteria
  - Cross validation
- Model averaging

# Model Comparison

- How can we compare two or more models for the same data?
  - We have used posterior predictive checks to assess how well a model explains the data used to fit a model.
  - We have looked at Bayesian p values.
  - Now we’ll learn additional methods for comparing between models.

# Model Comparison

- We aim for:
  - **High goodness of fit**
    - Model fits the data
  - **Lower complexity**
    - Fewer parameters (more parameters can lead to overfitting)
    - (Occam's razor)
  - **High generalizability**
    - Model predicts future data well

# Model Comparison

Important Terms:

- **Within-sample accuracy :**
  - The accuracy is measured with the same data used to fit the model.
- **Out-of-sample accuracy :**
  - The accuracy measured with data not used to fit the model.
- The within-sample accuracy will be higher and lead us to believe our model is better than it is.
- Leaving data out means less data to fit our model.

# Model Comparison

- To overcome this, we will use two methods:
  - **Information criteria:** Expressions that approximate out-of-sample accuracy as in-sample accuracy plus a term that penalizes model complexity.
  - **Cross-validation:** A method that involves dividing the available data into separate subsets that are alternatively used to fit and evaluate the models

# Widely Applicable Information Criteria

$$WAIC = -2 \underbrace{\sum_{i}^{n} \log\left(\frac{1}{S}\sum_{s=1}^{S} p(y_i \mid \theta^s)\right)}_{\text{Log pointwise predictive density (lppd)}} + 2 \underbrace{\sum_{i}^{n} \left(V_{s=1}^{S} \log p(y_i \mid \theta^s)\right)}_{\text{Penalty } P_{WAIC}}$$

(1) lppd - A measure of how well the model fits the data

(2) penalty $P_{WAIC}$ –

- Common interpretation - A measure of the effective number of parameters
- Actually - An estimate of how much worse the likelihood would be if we were looking at new data.

# Widely Applicable Information Criteria

$$WAIC = \underbrace{-2\sum_{i}^{n}\log\left(\frac{1}{S}\sum_{s=1}^{S}p(y_i \mid \theta^s)\right)}_{\text{Log pointwise predictive density (lppd)}} + \underbrace{2\sum_{i}^{n}\left(V_{s=1}^{S}\log p(y_i \mid \theta^s)\right)}_{\text{Penalty } P_{WAIC}}$$

- **We will choose the model with the lower WAIC.**
- If two models fit the data equally well, we will choose the simpler one.

# Cross Validation

- Divide our data into k parts.
- Use k-1 parts to fit the model and test it on the left-out portion.
- We get k models and k accuracy values.
- The accuracy of the model is the average of the k accuracy values.
- Fit the model on all the data one final time.
  - This is the final model for future use.

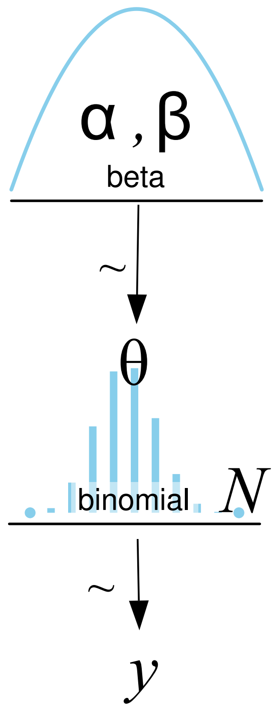
# Cross Validation

- When K equals the number of data points, we get what is known as **leave-one-out cross validation (LOOCV)**, meaning we fit the model to <u>all but one data point each time</u>.
- Practically, this is computationally expensive, and we are going to estimate it.

# Null and Alternative Model

- Null model: coin is fair.
- Alternative model: coin is biased.

**Graphical model**




**PyMC (a PPL)**

```
coords = {"data": np.arange(len(data))}

with pm.Model(coords=coords) as model_1:
    thet = pm.Beta('thet', alpha=1., beta=1.)
    y = pm.Bernoulli('y', p=thet, observed=data, dims = 'data')
    idata1 = pm.sample(1000, chains = 4, idata_kwargs={"log_likelihood":True})
```

arviz.InferenceData

- ▸ posterior
- ▸ log_likelihood
- ▸ sample_stats
- ▸ observed_data

# Null and Alternative Model

- First, let's look at the WAIC and LOO of our model:

```
az.waic(idata1)
```

| | 0 |
|---|---|
| **elpd_waic** | -60.260962 |
| **se** | 4.245041 |
| **p_waic** | 0.961681 |
| **n_samples** | 4000 |
| **n_data_points** | 100 |
| **warning** | False |
| **waic_i** | [<xarray.DataArray 'waic_i' ()> Size: 8B\narra... |
| **scale** | log |

```
print(az.loo(idata1))
```

```
Computed from 4000 posterior samples and 100 observations log-likelihood matrix.

         Estimate       SE
elpd_loo   -60.26     4.25
p_loo        0.96        -
------

Pareto k diagnostic values:
                         Count   Pct.
(-Inf, 0.70]   (good)      100  100.0%
   (0.70, 1]   (bad)         0    0.0%
   (1, Inf)   (very bad)     0    0.0%
```

Effective number of parameters

How reliable is our estimate?
Values above 0.7 indicate that we may have very influential datapoints – bad.
We can see that 100% of our datapoints are good.

- Now let's compare the values to those of the null model.

# az. compare

https://python.arviz.org/en/stable/api/generated/arviz.compare.html

**Returns:**

A DataFrame, ordered from best to worst model (measured by the ELPD).

The index reflects the key with which the models are passed to this function. The c

**rank:** The rank-order of the models. 0 is the best.

**elpd:** ELPD estimated either using (PSIS-LOO-CV elpd_loo or WAIC elpd_waic).

Higher ELPD indicates higher out-of-sample predictive fit ("better" model). If scale is de
negative_log smaller values indicates higher out-of-sample predictive fit ("better" model

**pIC:** Estimated effective number of parameters.

**elpd_diff:** The difference in ELPD between two models.

If more than two models are compared, the difference is computed relative to the top-ran
that always has a elpd_diff of 0.

**weight:** Relative weight for each model.

This can be loosely interpreted as the probability of each model (among the compared m
the data. By default the uncertainty in the weights estimation is considered using Bayesiar

**SE:** Standard error of the ELPD estimate.

If method = BB-pseudo-BMA these values are estimated using Bayesian bootstrap.

**dSE:** Standard error of the difference in ELPD between each model and the top-ranked

It's always 0 for the top-ranked model.

**warning:** A value of 1 indicates that the computation of the ELPD may not be reliable.

This could be indication of WAIC/LOO starting to fail see http://arxiv.org/abs/1507.04544

**scale:** Scale used for the ELPD.

# Null and Alternative Model

```
# Compare
az.compare({"alternative": idata1, "null": idata_null})
```

From best to worst

Higher = better

| | rank | elpd_loo | p_loo | elpd_diff | weight | se | dse | warning | scale |
|---|---|---|---|---|---|---|---|---|---|
| **alternative** | 0 | -60.263062 | 0.963782 | 0.000000 | 0.94854 | 4.245280 | 0.000000 | False | log |
| **null** | 1 | -65.995594 | 0.193007 | 5.732533 | 0.05146 | 0.800336 | 3.444944 | False | log |

If the difference between the elpd values is bigger

than the standard error

Then we say that there is a meaningful difference between the two models

# Null and Alternative Model

- That was the LOO values, now we can do the same with the widely accepted information criteria.

```
# Compare WAIC
az.compare({"alternative": idata1, "null": idata_null}, ic = 'waic')
```

| | rank | elpd_waic | p_waic | elpd_diff | weight | se | dse | warning | scale |
|---|---|---|---|---|---|---|---|---|---|
| **alternative** | 0 | -60.260962 | 0.961681 | 0.000000 | 0.948718 | 4.245041 | 0.000000 | False | log |
| **null** | 1 | -65.995353 | 0.192766 | 5.734392 | 0.051282 | 0.800333 | 3.444707 | False | log |

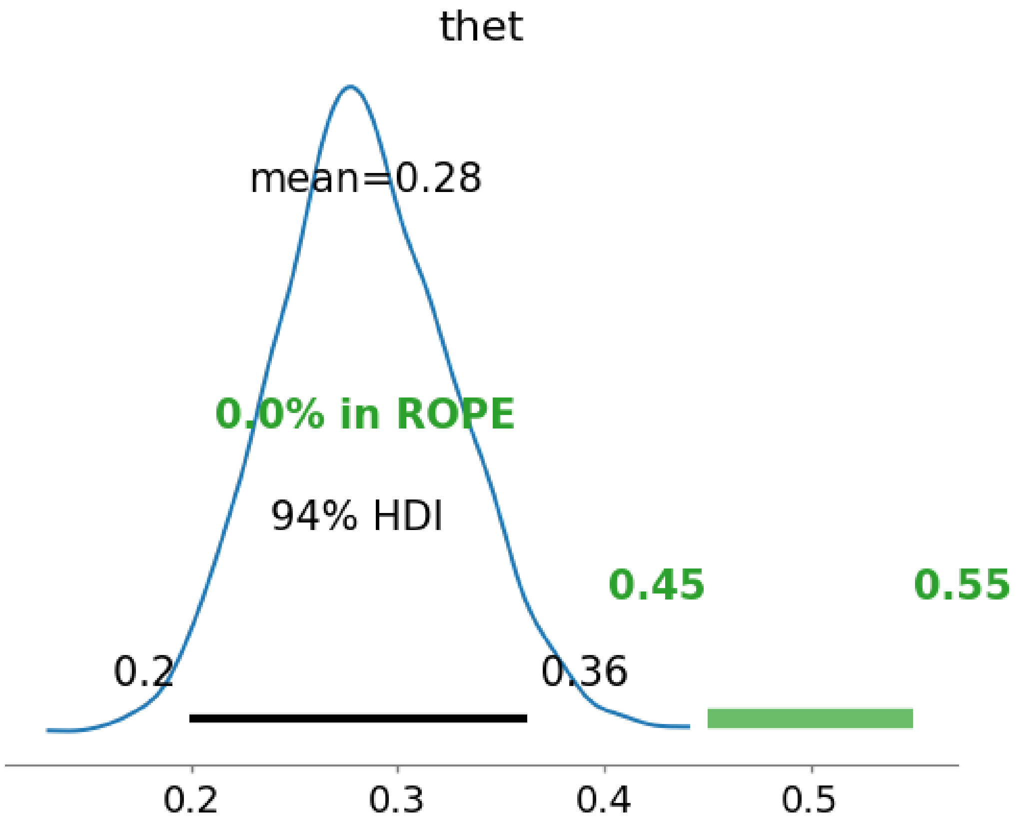
- The WAIC and LOO values are generally almost exactly the same.
  - At this point, it is generally more accepted to use the LOO.

# Null and Alternative Model

- We created the null model by using a very narrow posterior defined by the ROPE.
- We defined the ROPE as [0.45, 0.55].
- Another option for comparison is to check if the HDI and ROPE overlap.
- Based on our model comparison using LOO and WAIC, we concluded that there is a meaningful difference between the null and alternative models.

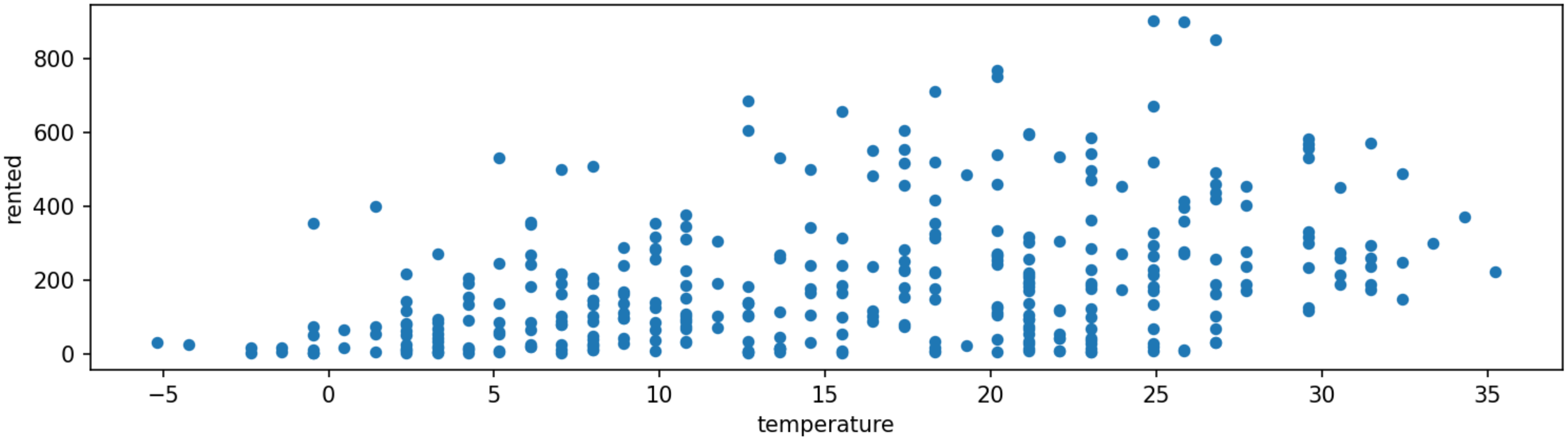
# Null and Alternative Model

- We arrive at the same conclusion looking at the HDI and ROPE
  - There is no overlap between them
  - We reject the null model of a fair coin



# Bikes Example

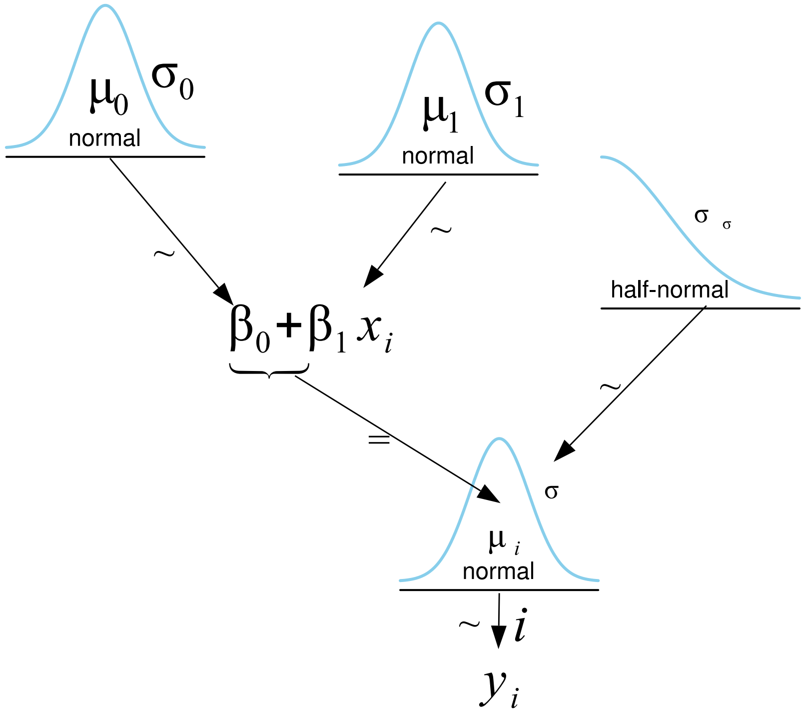
- In the lecture, you looked at bike rentals as a function of temperature using several models:



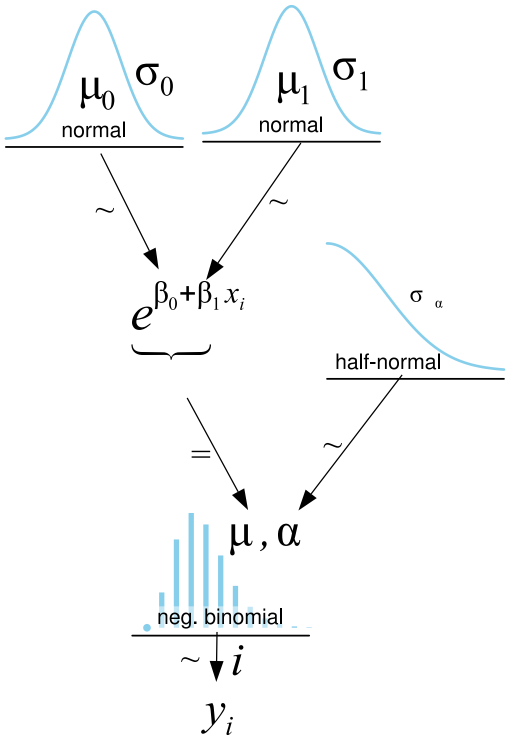
# Bikes Example

- In the lecture, you looked at bike rentals as a function of temperature using two models:

**Linear Model**




**Negative Binomial Model**



# Bikes Example

- In the lecture, you looked at bike rentals as a function of temperature using two models:

**Linear Model**

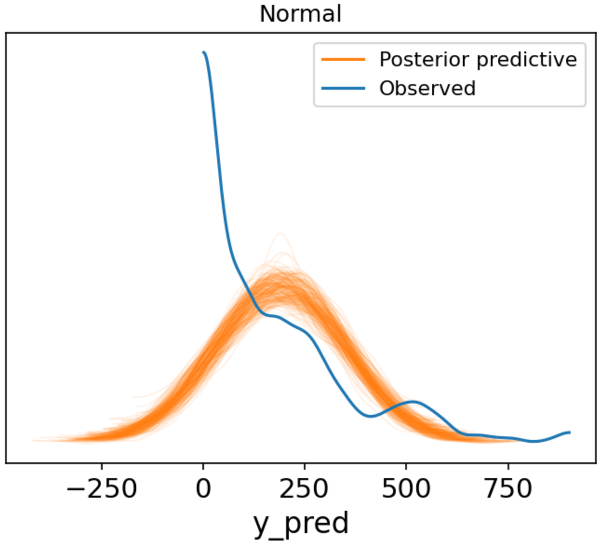
```
coords = {"data": np.arange(len(bikes))}
with pm.Model(coords=coords) as model_lb:
    β0 = pm.Normal("β0", mu=0, sigma=100)
    β1 = pm.Normal("β1", mu=0, sigma=10)
    σ = pm.HalfNormal("σ", 10)
    μ = pm.Deterministic("μ", β0 + β1 * bikes.temperature, dims="data")
    y_pred = pm.Normal("y_pred", mu=μ, sigma=σ, observed=bikes.rented, dims="data")
    idata_lb = pm.sample(1000, chains = 4, idata_kwargs={"log_likelihood":True})
```

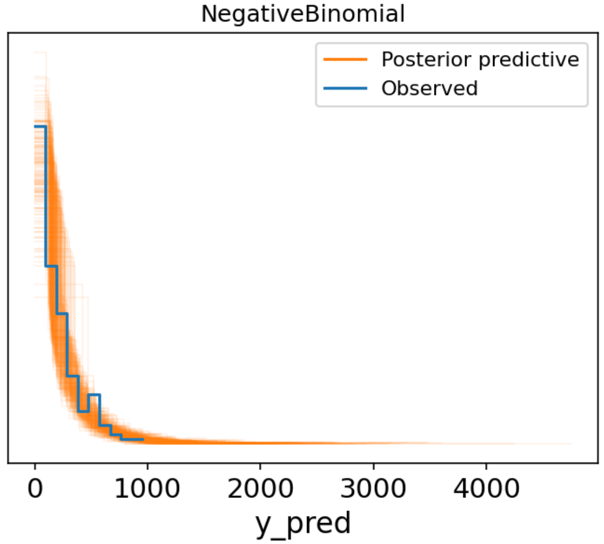
**Negative Binomial Model**

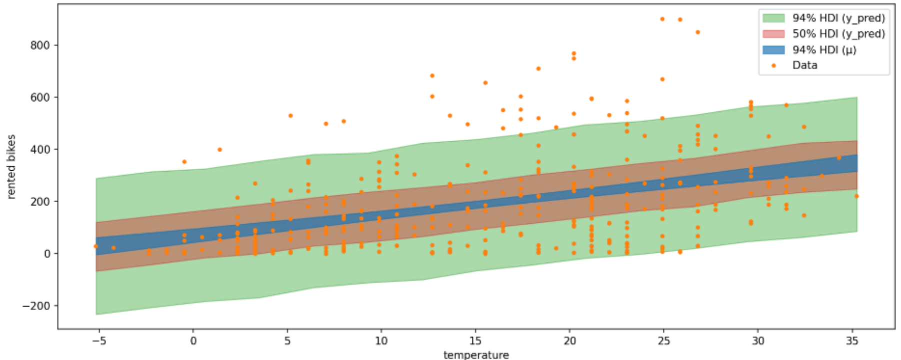
```
with pm.Model() as model_neg:
    beta0 = pm.Normal("beta0", mu=mu_0, sigma=sigma_0)
    beta1 = pm.Normal("beta1", mu=mu_1, sigma=sigma_1)
    alpha = pm.HalfNormal("alpha", sigma=sigma_alpha)
    mu = pm.Deterministic("mu", pm.math.exp(beta0 + beta1 * bikes.temperature))
    y_pred = pm.NegativeBinomial("y_pred", mu=mu, alpha=alpha, observed=bikes.rented)
    idata_neg = pm.sample(1000, chains = 4, idata_kwargs={"log_likelihood":True})
```

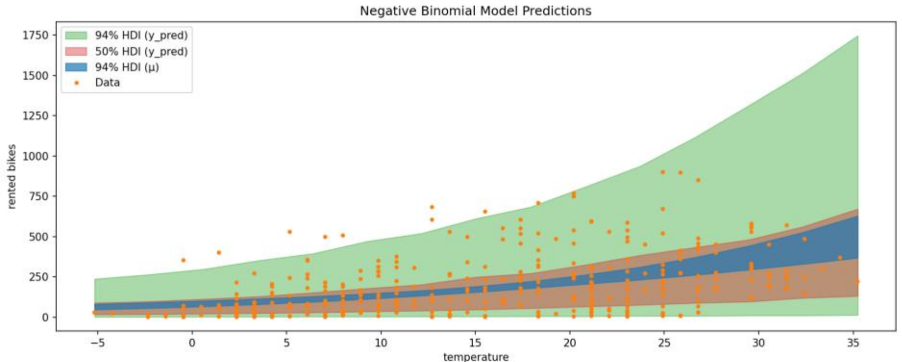
# Bikes Example

- In the lecture you compared the two using posterior predictive checks and saw that the negative binomial model was better:



# Bikes Example

- Let’s add the WAIC and LOO to the comparison:

| | rank | elpd_loo | p_loo | elpd_diff | weight | se | dse | warning | scale |
|---|---|---|---|---|---|---|---|---|---|
| **negative_binomial** | 0 | -2153.169315 | 2.742507 | 0.000000 | 1.0 | 19.804978 | 0.000000 | False | log |
| **linear** | 1 | -2300.275960 | 4.858970 | 147.106645 | 0.0 | 26.995013 | 21.713316 | False | log |

| | rank | elpd_waic | p_waic | elpd_diff | weight | se | dse | warning | scale |
|---|---|---|---|---|---|---|---|---|---|
| **negative_binomial** | 0 | -2153.163883 | 2.737075 | 0.000000 | 1.0 | 19.804702 | 0.000000 | False | log |
| **linear** | 1 | -2300.267411 | 4.850421 | 147.103528 | 0.0 | 26.993311 | 21.711993 | True | log |

# Bikes Example

- We also learned about multiple regression.
- Another use for model comparison can be to test the value of adding additional independent variables.
  - Adds information to the model.
  - Adds complexity.
- We can add another independent variable of the humidity of the day.

# Bikes Example

- Create and sample:

```
with pm.Model() as model_mlb:
    α = pm.Normal("α", mu=0, sigma=1)
    β0 = pm.Normal("β0", mu=0, sigma=10)
    β1 = pm.Normal("β1", mu=0, sigma=10)
    σ = pm.HalfNormal("σ", 10)
    μ = pm.Deterministic("μ", pm.math.exp(α + β0 * bikes.temperature + β1 * bikes.humidity))
    _ = pm.NegativeBinomial("y_pred", mu=μ, alpha=σ, observed=bikes.rented)

    idata_mlb = pm.sample(1000, chains = 4, idata_kwargs={"log_likelihood":True})
```

# Bikes Example

- Now let's compare all three:

```
az.compare({"linear": idata_lb, "negative_binomial": idata_neg, "negative_binomial_multiple": idata_mlb})
```

| | rank | elpd_loo | p_loo | elpd_diff | weight | se | dse | warning | scale |
|---|---|---|---|---|---|---|---|---|---|
| **negative_binomial_multiple** | 0 | -2141.148332 | 3.725948 | 0.000000 | 1.000000e+00 | 20.860020 | 0.000000 | False | log |
| **negative_binomial** | 1 | -2153.169315 | 2.742507 | 12.020983 | 0.000000e+00 | 19.804978 | 3.837865 | False | log |
| **linear** | 2 | -2300.275960 | 4.858970 | 159.127628 | 2.428169e-11 | 26.995013 | 22.536404 | False | log |

- We can see that the difference between the LOO values for the two negative binomial models is larger than the standard error.
  - We can conclude from this that it is worth adding the humidity despite it leading to a more complex model.

# When can’t we trust WAIC?

- WAIC is based on a Taylor expansion of the log-pointwise predictive density (lpd).
  - If the estimated effective number of parameters (p_waic) grows too large relative to your sample

$$\frac{P_{waic}}{N} > 0.4$$

  - the variance correction becomes unreliable.
- Symptoms:
  - WAIC’s standard error (se_waic) spikes
  - Posterior draws give wildly different WAIC values

**warning:**bool

True if posterior variance of the log predictive densities exceeds 0.4

https://python.arviz.org/en/latest/api/generated/arviz.waic.html

# When can’t we trust LOO?

- LOO uses Pareto-smoothed importance sampling to approximate refits.
- Key diagnostic: the Pareto shape parameters *k* morf loo.pareto_k.

- Rule of thumb:
  - If any *K  > 0.1* sampling variance infinite
    → **don’t trust LOO**
  - If more than ~5% of points have K > 0.7
    →**beware**

- Symptoms:
  - LOO se is huge
  - Repeated reruns give different ranking

# When can't we trust LOO?

**Returns:**

`ELPDData` `object` ( `inherits` `from` `pandas.Series` ) `with` `the` `following` **row/attributes:**

**elpd_loo:** `approximated` `expected` `log` `pointwise` `predictive` `density` ( `elpd` )

**se:** `standard` `error` **of** `the` `elpd`

**p_loo:** `effective` `number` **of** `parameters`

**n_samples:** `number` **of** `samples`

**n_data_points:** `number` **of** `data` `points`

**warning:**bool

True if the estimated shape parameter of Pareto distribution is greater than `good_k`.

**loo_i:** `DataArray` `with` `the` `pointwise` `predictive` **accuracy,**

only if pointwise=True

**pareto_k:** `array` **of** `Pareto` `shape` `values` , `only` `if` `pointwise` `True`

**scale:** `scale` **of** `the` `elpd`

**good_k:** `For` `a` `sample` `size` **S,** `the` `thresold` `is` `compute` `as` `min` **(1 - 1/log10(S), 0.7)**

The returned object has a custom print method that overrides pd.Series method.

https://arxiv.org/abs/1507.04544 https://arxiv.org/abs/1507.02646

# Model Averaging

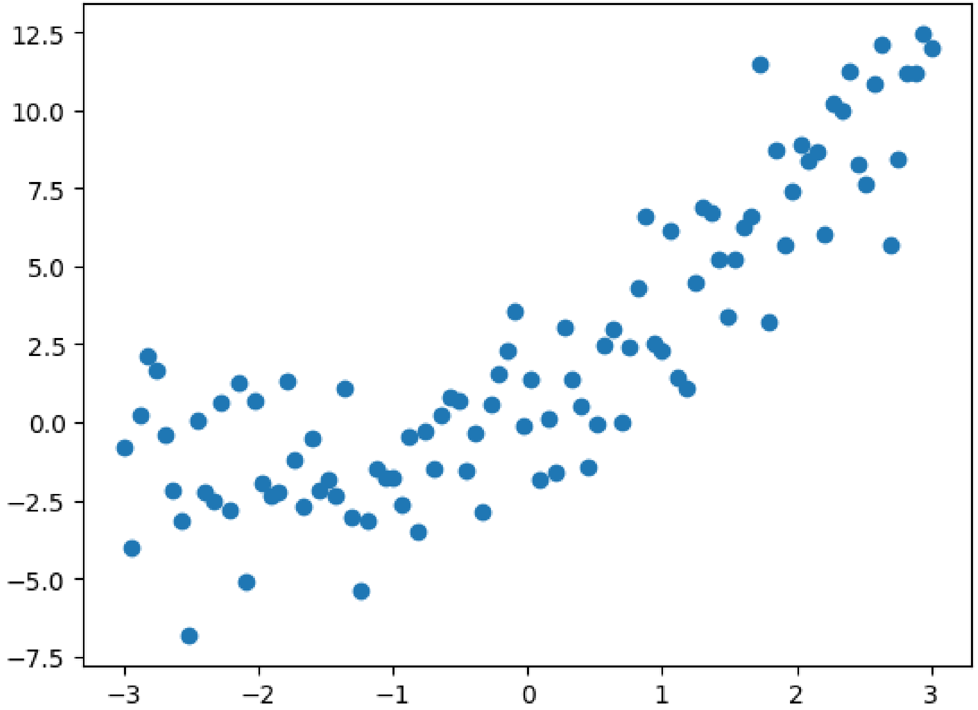
- Instead of choosing one model, we can also average the different models.
- We can compute a weighted average of the different models.
- The weights are computed in the compare function.
  - They are the relative weight of each model and, in large, represent the probability of each model given the data.

```
az.compare({"linear": idata_lb, "negative_binomial": idata_neg, "negative_binomial_multiple": idata_mlb})
```

| | rank | elpd_loo | p_loo | elpd_diff | weight | se | dse | warning | scale |
|---|---|---|---|---|---|---|---|---|---|
| **negative_binomial_multiple** | 0 | -2141.148332 | 3.725948 | 0.000000 | 1.000000e+00 | 20.860020 | 0.000000 | False | log |
| **negative_binomial** | 1 | -2153.169315 | 2.742507 | 12.020983 | 0.000000e+00 | 19.804978 | 3.837865 | False | log |
| **linear** | 2 | -2300.275960 | 4.858970 | 159.127628 | 2.428169e-11 | 26.995013 | 22.536404 | False | log |

# Model Averaging

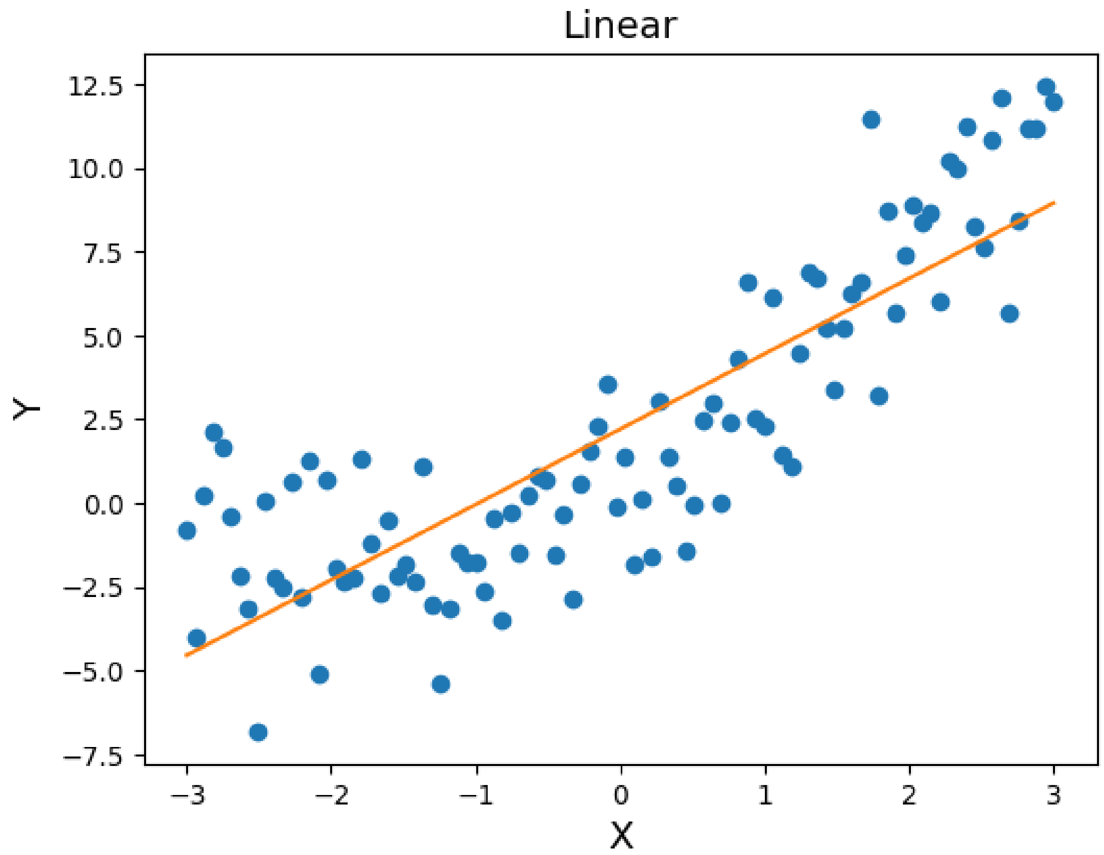
- Let’s create random data with two models that give not zero weights.

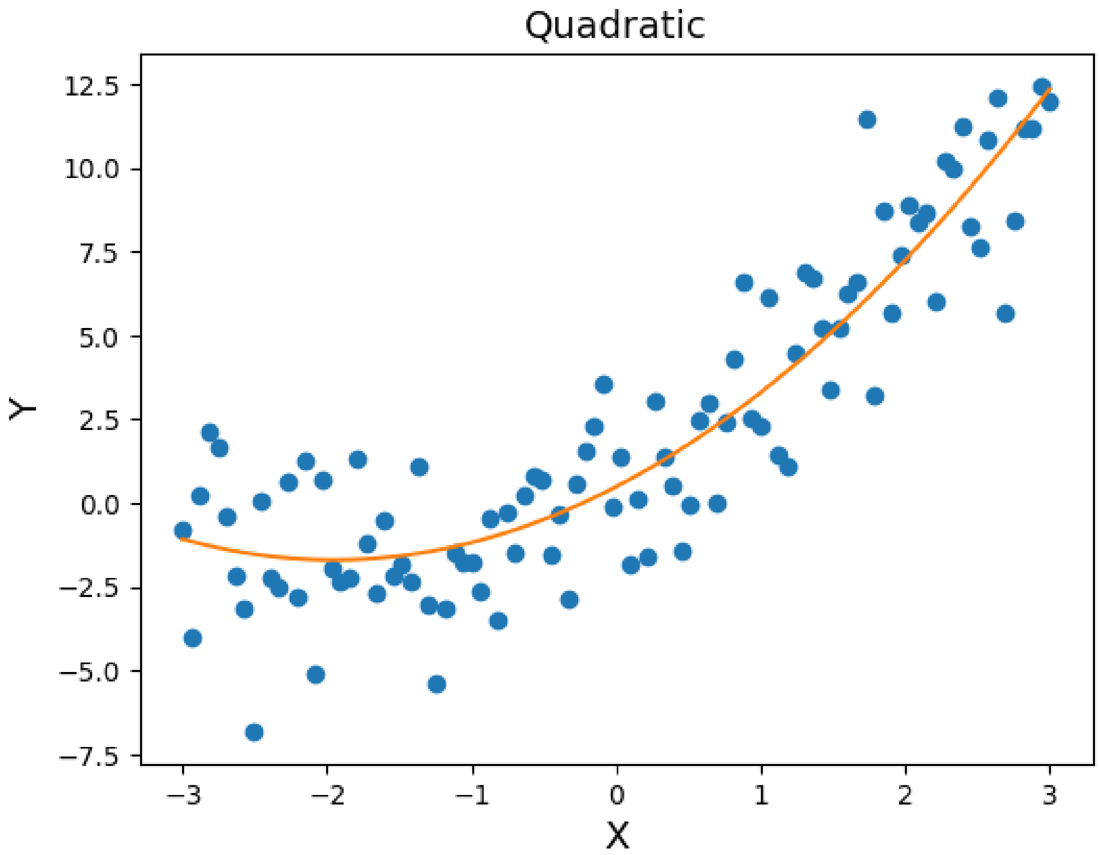



- We’ll use a linear model
  - Simpler but fits data less well
- And a quadratic model
  - Fits better but more complex

# Model Averaging

- The means of the two models:



# Model Averaging

- Comparing between them yields:

```
cmp_df = az.compare({"linear": idata_linear, "quadratic": idata_quad})
cmp_df
```

| | rank | elpd_loo | p_loo | elpd_diff | weight | se | dse | warning | scale |
|---|---|---|---|---|---|---|---|---|---|
| **quadratic** | 0 | -212.428280 | 4.036177 | 0.000000 | 0.945733 | 7.300654 | 0.000000 | False | log |
| **linear** | 1 | -236.306868 | 2.944081 | 23.878587 | 0.054267 | 6.160884 | 6.660192 | False | log |

- We will use these weights to compute the weighted average model.

# Model Averaging

- Comparing between them yields:

```
cmp_df = az.compare({"linear": idata_linear, "quadratic": idata_quad})
cmp_df
```

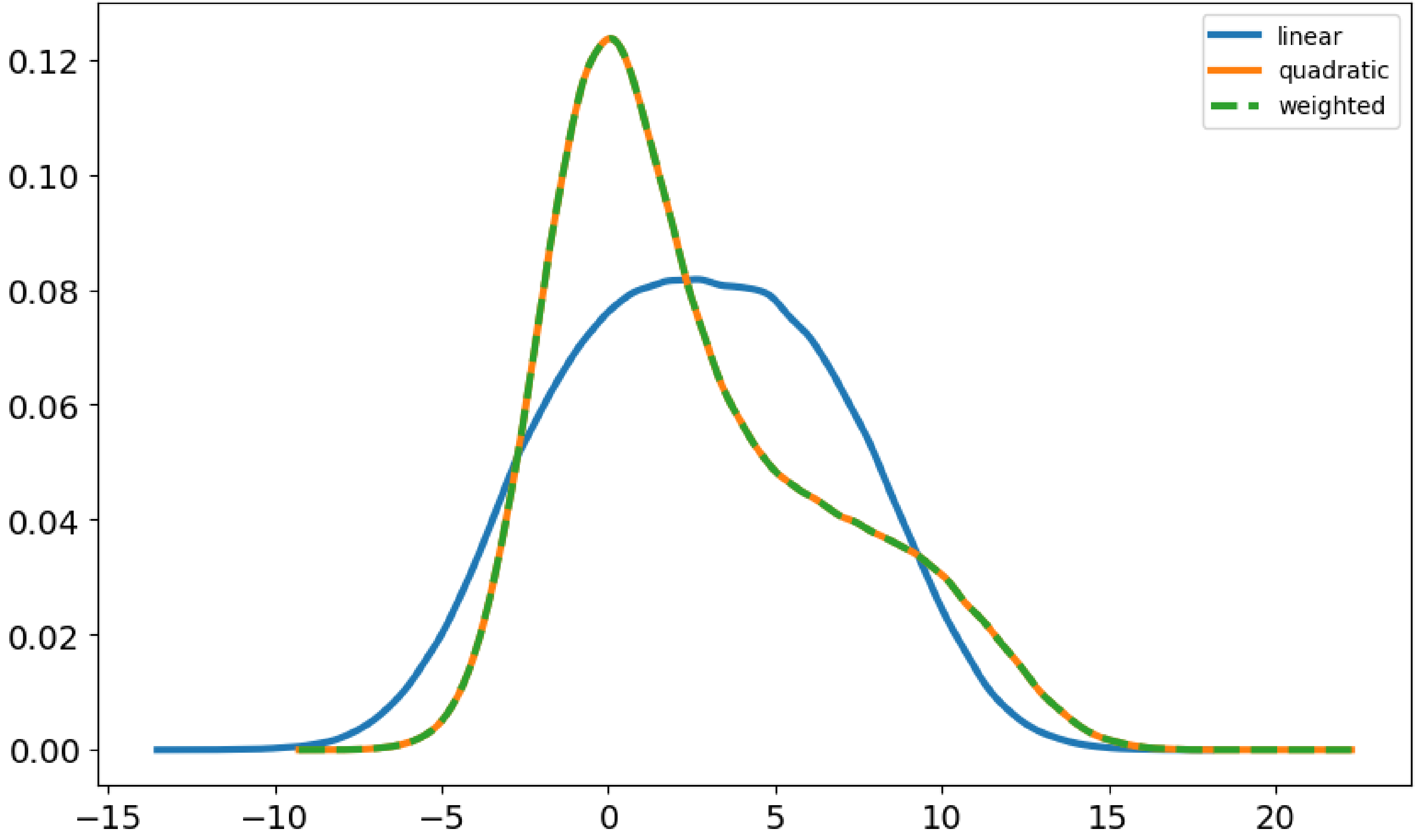
| | rank | elpd_loo | p_loo | elpd_diff | weight | se | dse | warning | scale |
|---|---|---|---|---|---|---|---|---|---|
| **quadratic** | 0 | -212.428280 | 4.036177 | 0.000000 | 0.945733 | 7.300654 | 0.000000 | False | log |
| **linear** | 1 | -236.306868 | 2.944081 | 23.878587 | 0.054267 | 6.160884 | 6.660192 | False | log |

- We will use these weights to compute the weighted average model.

```
avg_preds = az.weight_predictions([idata_quad, idata_linear], weights=cmp_df["weight"].values)
```

# Model Averaging

- Plotting:



# Model Averaging

- We can also define the weights however we like.
  - Example: half to each of the two models.