Project 1 - Linear Regression

CS146 - Computational Methods for Bayesian Statistics
Thu Than

October 31, 2024

Contents

1 Data Description

Argentina is one of the most baffling economies in the world, characterized by soaring inflation rates and a booming black market for dollars. There are many interesting economic factors to analyze about Argentina. In this report, I want to explore how government expenditure affects the country's real gross domestic product.

General government expenditure includes all government expenditures for purchases of goods and services (including compensation of employees). **Gross domestic product (GDP)** is the standard measure of the value added created through the production of goods and services in a country. GDP, therefore, measures the income earned from that population or the total amount spent on final goods and services.

Government expenditure is the **predictor (x)** and gross domestic product is the **outcome (y)**. The two variables are measured in **millions of Argentina pesos**. All observations are recorded **quarterly in 10 years, from Q1/2004 to Q1/2024**. There are a total of **81 values** in the dataset. All code can be found in the Code Appendix section.

2 Models

Given that government expenditure and GDP are continuous variables, linear regression can be used as a predictive model for the dataset. According to Keynesian economics, increased government spending raises aggregate demand and increases consumption, which leads to increased production. Therefore, the two variables are very likely to co-vary in a positively linear way.

Before diving in, all models' samplers work well because all chains' distributions follow a uniform distribution, indicating that the sampler explores the full posterior distribution. The metric r_hat of all parameters is 1.0, and the effect sizes are around a few hundred to thousands. The code and figures of these diagnostics can be found in the appendix.

2.1 Normal Model

Likelihood:

$$y_i \sim \text{Normal}(\mu_i, \sigma^2)$$

$$\mu_i = c_0 + c_1 x_i + c_2 x_i^2 + ... + c_k x_i^k$$

Prior:

$$c_0 \sim \text{Uniform}(0, 20)$$

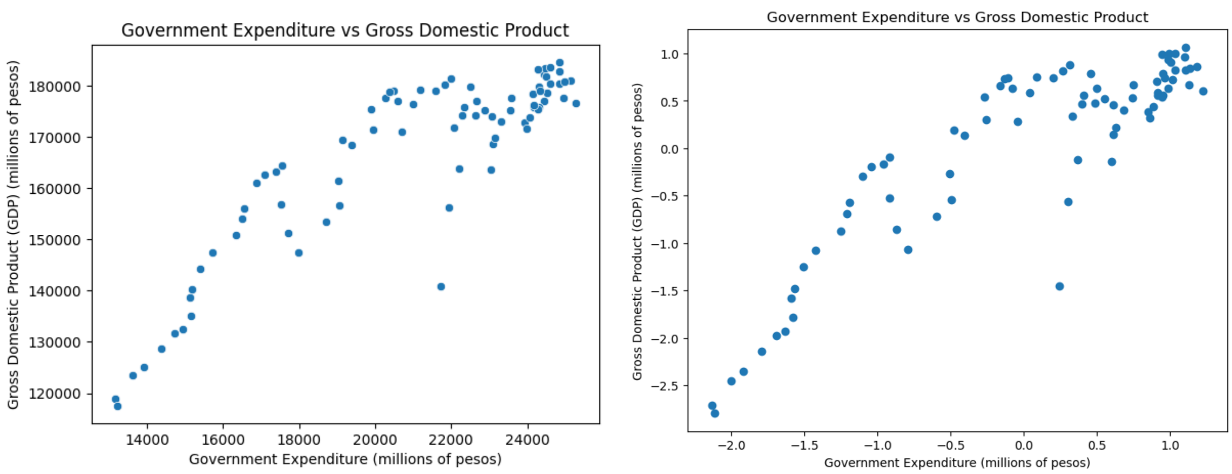
$$c_k \sim \text{Normal}(0, 1)$$

$$\sigma \sim \text{Half-Normal}(0, 5)$$

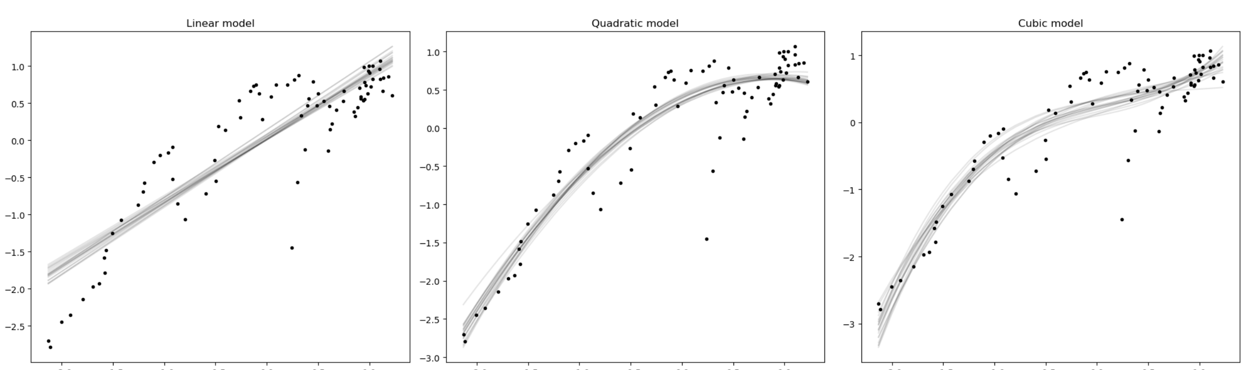
Priors are the distribution over the parameters, which are chosen based on our current knowledge about the subject. For the intercept c_0, which indicates the GDP when government spending equals zero, I chose a **Uniform distribution with the lower bound of 0 and an upper bound of 20**. This is because the variables are already measured in millions of pesos. I argue that the values can only be within the range of [0, 20]. Theoretically, GDP can take negative values, but because this occasion is infrequent (e.g., a recession or economic crisis), I decided not to use a

Normal distribution (which allows negative values) in this case.

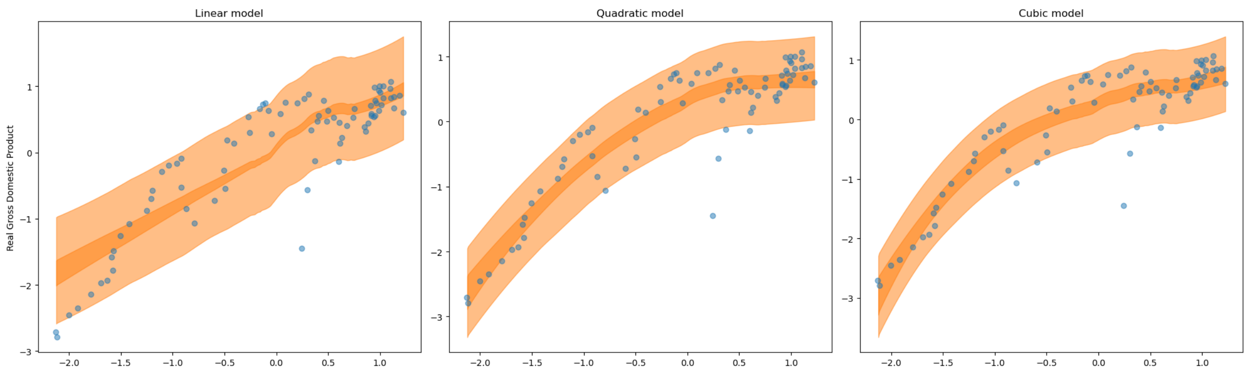
For the coefficients c, I chose a **Normal distribution with a mean of 0 and a standard deviation of 1**. This is a relatively neutral prior on the effect of government spending on GDP. Despite my earlier argument that increased spending is associated with increased GDP, it is possible that the reverse is also true. According to the National Bureau of Economic Research, increases in public expenditure can hit company profits and thus lead to a reduction in private investment and economic growth. We are talking about economics, not natural measurements such as weights or heights, so negative values should be considered rather than completely excluded. For the standard deviation of the outcome, I chose **a Half-Normal distribution with mean 0** because standard deviation cannot take on negative values, and to allow for enough possibilities, I decided to choose **sigma equal 5**.



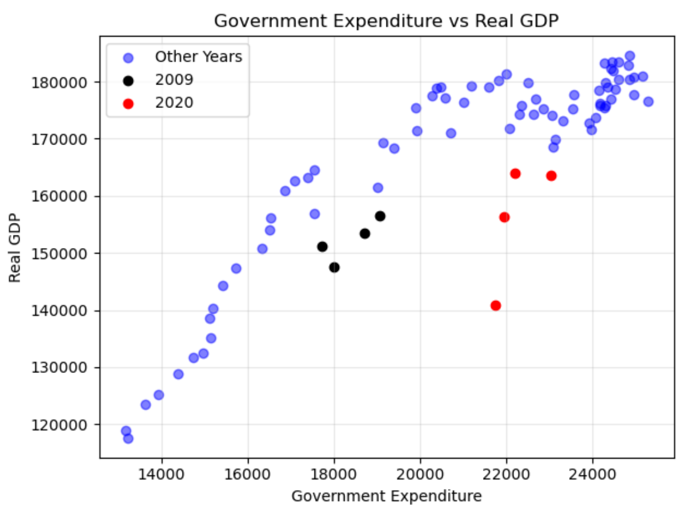
In the scatter plots above, the relationship between GDP and government expenditure does not follow a straight line, but it is not overly complex either. As a result, a polynomial regression is suitable for this non-linear data. To decide which degree best fits the data, we will perform a model comparison of Normal likelihood with different degrees of 1, 2, and 3. Before fitting the linear model, I standardized the data so that its mean was centered at zero with one standard deviation. This ensures that the values are rescaled to be in the range of [-1, 1] to work better with the sampling process.



Three figures above represent the **posterior distributions** of Normal linear models with different degrees. Visually, it is already clear that the model with a degree of 3 best fits the data, while the linear model with a degree of 1 underfits the data.



Posterior-predictive distributions of Normal linear models with different degrees and 89% highest density interval (the orange credible intervals). The linear model's interval captures almost all data points, meaning the model "thinks" most of them are plausible. In the quadratic and cubic normal models, 4 data points between 0 and 0.5 on the x-axis are outside the interval, with less probable true values. However, these models are still problematic. Why?



Let's reinvestigate the original dataset again because there is one special detail about its context: **2009 is during the Great Recession, and 2020 is the year of COVID-19**. This explains why, for the first two quarters of 2009, despite a slight increase in government spending, the GDP decreased before gradually improving in the next two quarters. Similarly, government spending increased in 2020, but because all economic activities were shut down during COVID-19, the GDP decreased compared to the previous years. **These occasions are rare**, so they are qualified as outliers. The posterior-predictive distribution of the Normal linear models shows that the models are affected by the first cluster of outliers because the four values of 2009 are within the 89% HDI. The models are less affected by the second cluster of outliers, but one dot is still very close or lies within the interval for all three models.

2.2 Student T Model

Likelihood:

$$y_i \sim T(\nu, \mu_i, \sigma)$$

$$\mu_i = c_0 + c_1 x_i + c_2 x_i^2 + ... + c_k x_i^k$$

Prior:

$$c_0 \sim \text{Uniform}(0, 20)$$

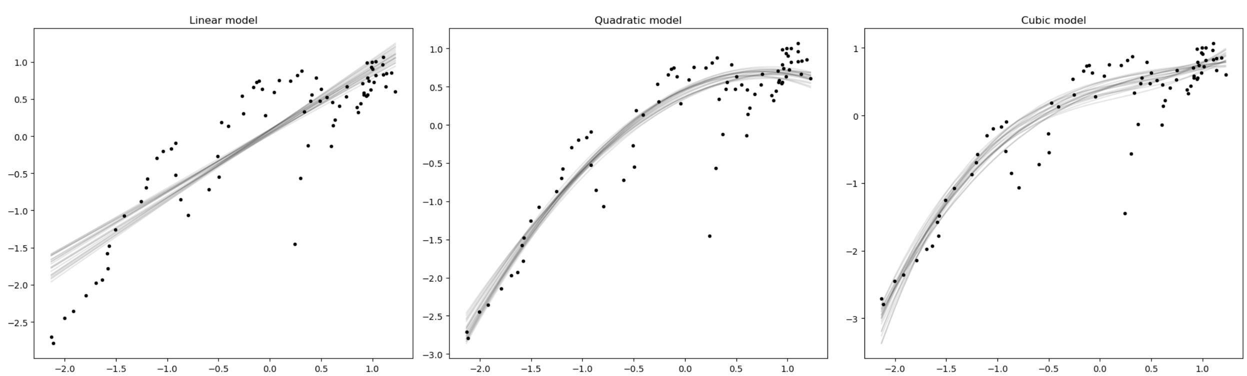
$$c_k \sim \text{Normal}(0, 1)$$

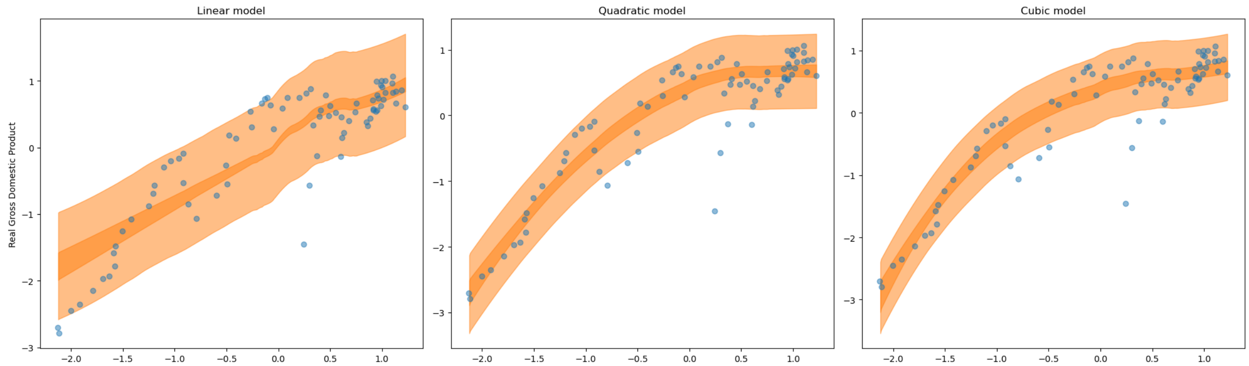
$$\sigma \sim \text{Half-Normal}(0, 5)$$

$$\nu \sim \text{Half-Normal}(\sigma = 30)$$

To make our model more robust to outliers (i.e., less sensitive to outliers so that it does not pull the line of best fit downward or upward), we will try another likelihood called student's t-distribution. Student's t-distribution has a special property in that it has a heavier tail, which assigns higher probabilities to extreme values. In other words, the model can accommodate outliers without affecting the overall fit.

I chose the same priors for the intercept and coefficients for the same reasons explained earlier. Student's t-distribution has another parameter ν, which determines the degrees of freedom. In this model, I chose Half-Normal distribution for ν because it has to be non-negative, with **mean zero and standard deviation (scale) of 30**. This is a reasonable scale (it still has heavier tails compared to the normal one) because outliers do not occur very often, which we need to account for in the dataset (i.e., there are only two rare occasions, which consists of 8 outliers). Extreme economic occasions only happened once in a while throughout history. As a result, 30 is a good balance for robustness and variability.





So, the posterior distributions of student t models are pretty much the same as those of the normal models. The critical difference lies in the posterior-predictive distribution. The linear model still performed quite badly because it included outliers as highly plausible. The quadratic and cubic models performed way better because the two clusters of outliers are clearly outside of the 89% credible intervals. The cubic model is the best because, in the quadratic one, two points of the first cluster of outliers are still close or inside the interval, which is not the case in the cubic one. This means the cubic model is not heavily affected by outliers, resulting in a more unbiased fit and estimates.

2.3 Outlier Detection

Likelihood:

$$y_i \sim \text{Normal}(\mu_i, \sigma_i)$$

$$\mu_i = c_0 + c_1 x_i + c_2 x_i^2 + c_3 x_i^3$$

$$\sigma_i = \begin{cases} \sigma_{\text{in}} & \text{if } q_i = 0 \\ \sigma_{\text{in}} + \sigma_{\text{out}} & \text{if } q_i = 1 \end{cases}$$

$$q \sim \text{Bernoulli}(p)$$

Prior:

$$c_0 \sim \text{Uniform}(0, 10)$$

$$c_1 \sim \text{Normal}(0, 1)$$

$$c_2 \sim \text{Normal}(0, 1)$$

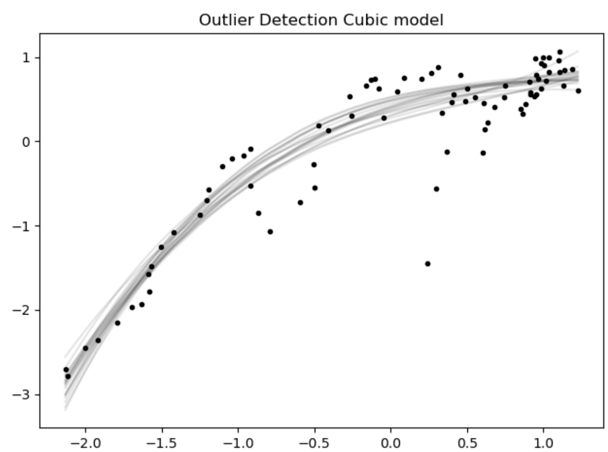
$$c_3 \sim \text{Normal}(0, 1)$$

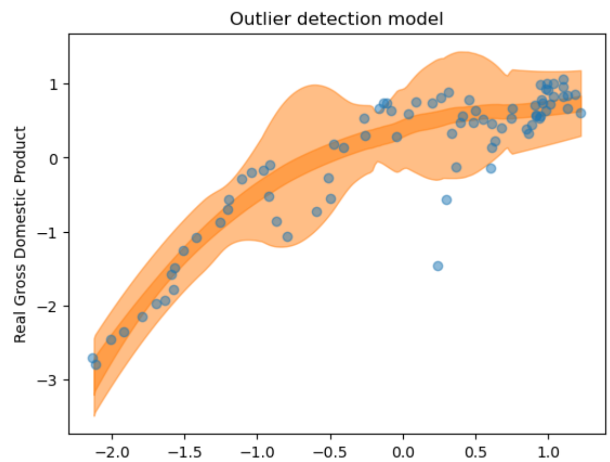
$$\sigma_{in} \sim \text{Half-Normal}(0, 10)$$

$$\sigma_{out} \sim \text{Half-Normal}(0, 30)$$

$$p \sim \text{Uniform}(0, 0.2)$$

For the outlier detection model, I assume that the standard deviation of each observation is not the same. It depends on another parameter that follows a Bernoulli distribution - 0 is the inliner, and 1 is the outlier. For the prior distribution over the intercept and the coefficients, I used identical distributions with the same reasoning for the Normal and Student-T distribution. Also, based on the two models above, I know that a cubic model is best at fitting the data due to its non-linear nature so I will use the same degree for the outlier detection. For the standard deviations of inliners and outliers, I chose the half-normal distribution for both because they cannot take on negative values. The former has a standard deviation of 10 because I assume that inliners do not vary too much and should cluster around the mean. The latter has a larger standard deviation of 30 because outliers might have more considerable variation and lie far away from the mean. I chose the **Uniform distribution with the lower bound of 0 and upper bound of 0.2** because, based on the current knowledge, the probability of the outliers (decreased GDP and increased spending) is small (i.e., over the past 15 years, there are only two signification events that slowed down the global economy: the Great Recession and the COVID-19 pandemic).





The left figure shows the posterior distribution, while the right figure shows the posterior-predictive distribution of the model. It is interesting that the model's credible interval captures the 2009 outliers but it is not affected by the 2020 outliers. This makes sense because, in this model, we used variance as the criteria to classify outliers. As a result, because the 2009 outliers are closer to the mean of the outcome, the model considers it as an inline despite the fact that it is actually "outlier" in real life. But for the 2020 outliers, their values are really far from the mean, which reflects in its variance, the model was not sensitive to them. In the model comparison section below, we can see that this model performs only slightly better than the student-t cubic model.

3 Model Comparison

To quantify which model(s) perform better, we will use two metrics: **the expected log-pointwise predictive density and the widely applicable information criteria**. Both use leave-one-out cross-validation to predict one missing data point at a time (i.e., we need to fit the model N times where N is the total number of data points) and average out the performance. Simply put, a model perfomes better than the other if its out-of-sample prediction is better. While the in-sample prediction performance is also important, focusing on it only is misleading because if we overfit the model, it can get perfect accuracy on the in-sample data (training data). At the end of the day, if our goal is to make good predictions, we need to focus on its ability to predict unseen data.

$$\text{elpd} = \text{expected log pointwise predictive density for a new dataset}$$

$$= \sum_{i=1}^{n} \int p_t(\tilde{y}_i) \log p(\tilde{y}_i|y) d\tilde{y}_i$$

$$\text{elpd}_{\text{loo}} = \text{elpd estimate of the leave-one-out cross-validation}$$
$$= \sum_{i=1}^{n} \log p(y_i|y_{-i})$$

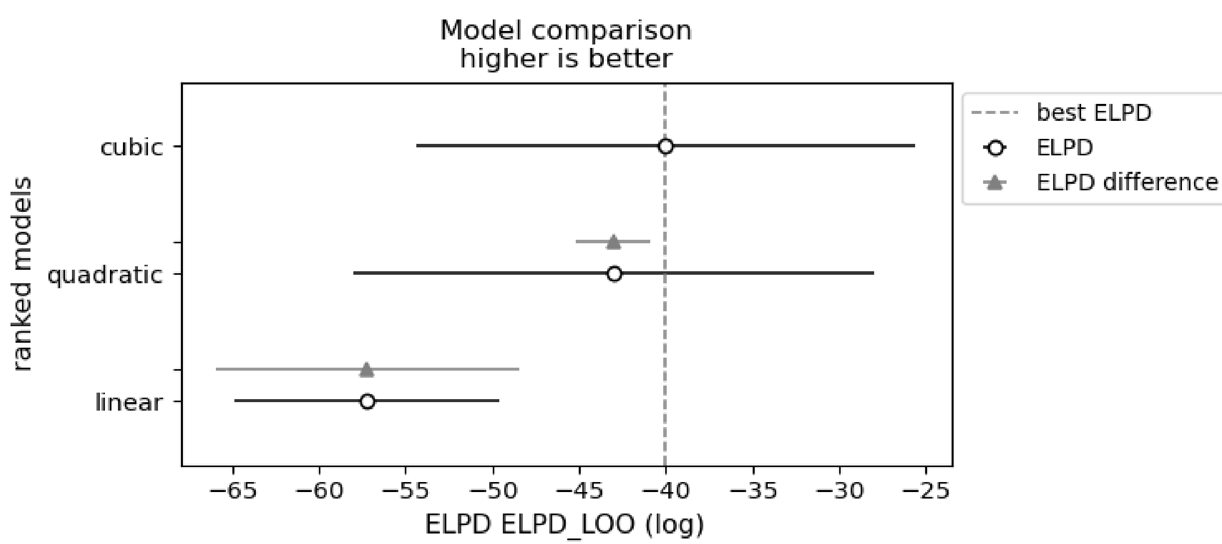
Widely Applicable Information Criteria:

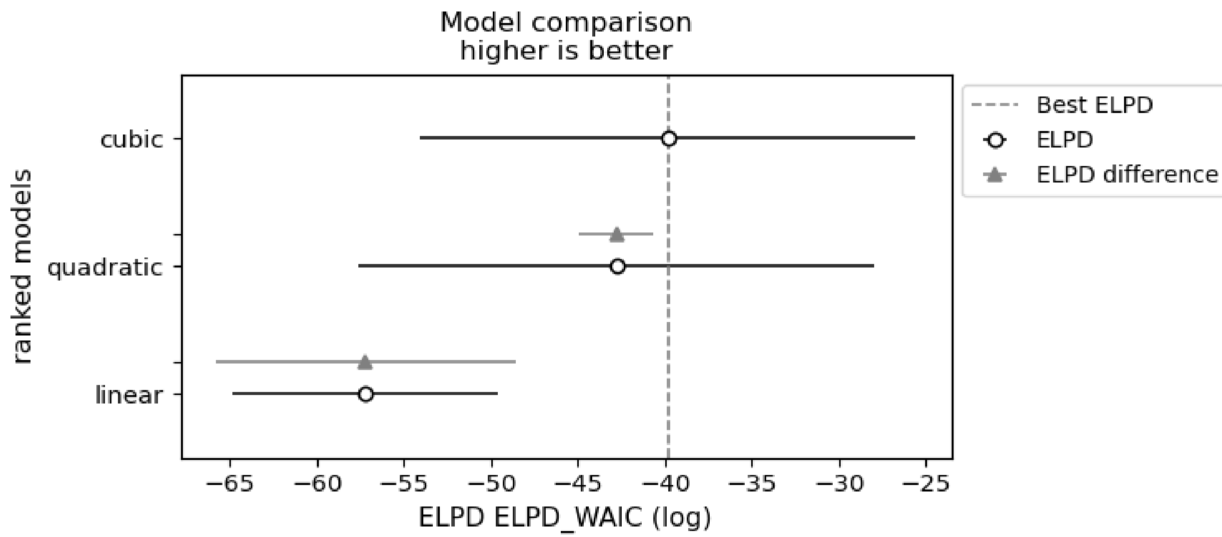
$$\hat{\text{elppd}}_{\text{WAIC}} = \text{lppd} - p_{\text{WAIC}}$$

$$\text{lppd} = \text{log pointwise predictive density} = \sum_{i=1}^{n} \log(\frac{1}{S} \sum_{s=1}^{S} p(y_i|\theta^s))$$

$$p_{\text{WAIC}} = 2 \sum_{i=1}^{n} (\log(\frac{1}{S} \sum_{s=1}^{S} p(y_i|\theta^s)) - \frac{1}{S} \sum_{s=1}^{S} \log p(y_i|\theta^s))$$

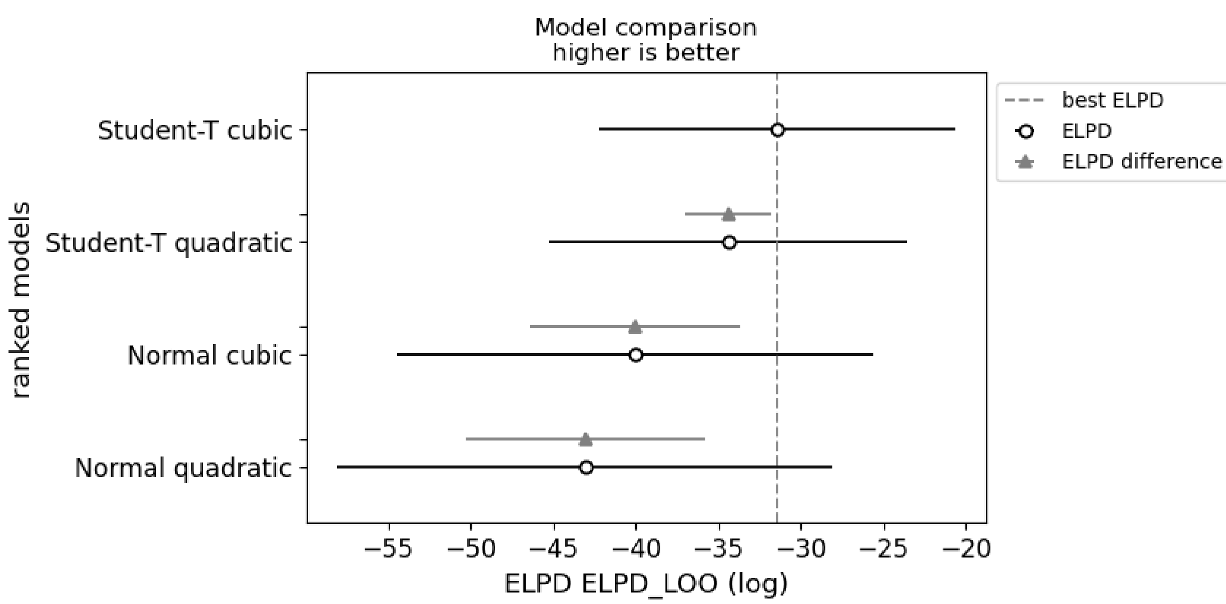
3.1 Normal Models

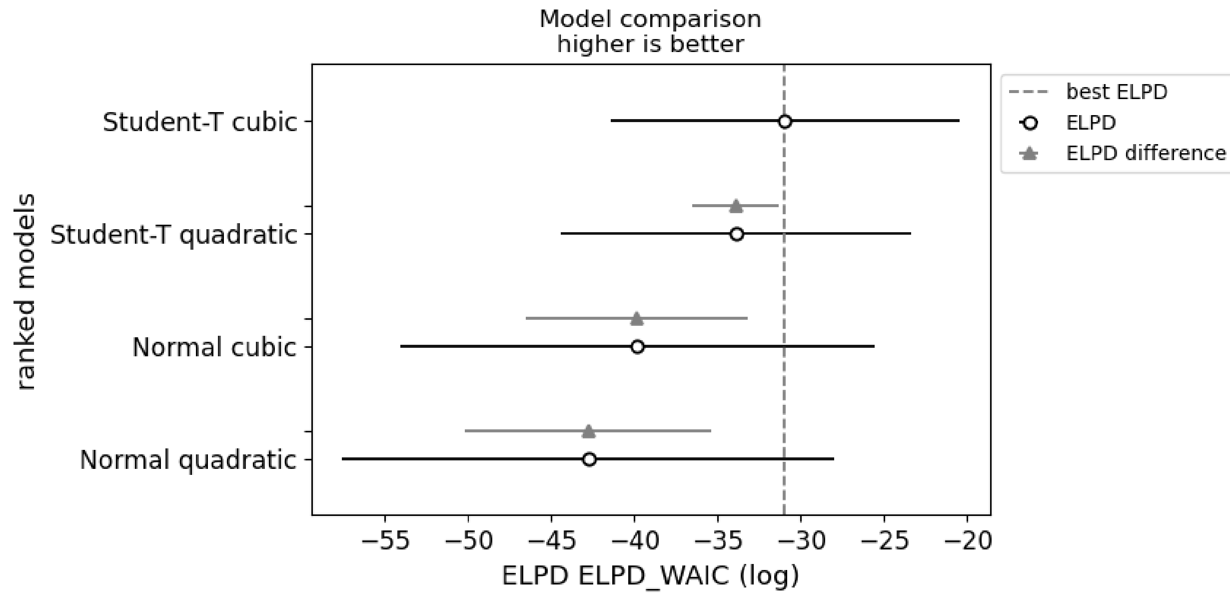




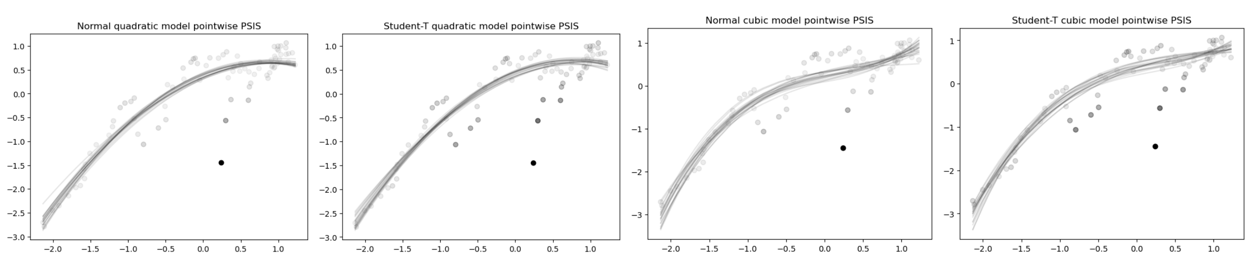
Based on the graphs, it is very clear that the cubic normal linear model outperformed other models in both metrics. If we look at the ELPD differences relative to the best model and the error bars, the error bars of the quadratic and the linear model do not overlap with the best ELPD. Therefore, we are pretty confident to conclude that relative to the linear and quadratic, the cubic performed best. More specifically, **the ELPD_LOO of the cubic model is roughly -40 compared to -43 (quadratic) and -57 (linear)**.

3.2 Normal vs. Student-T Models



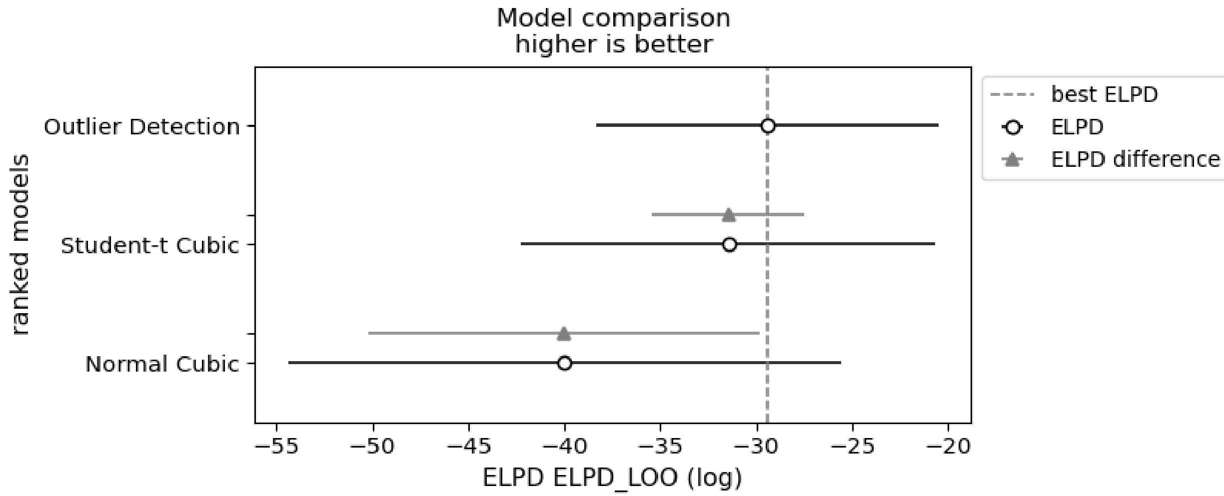


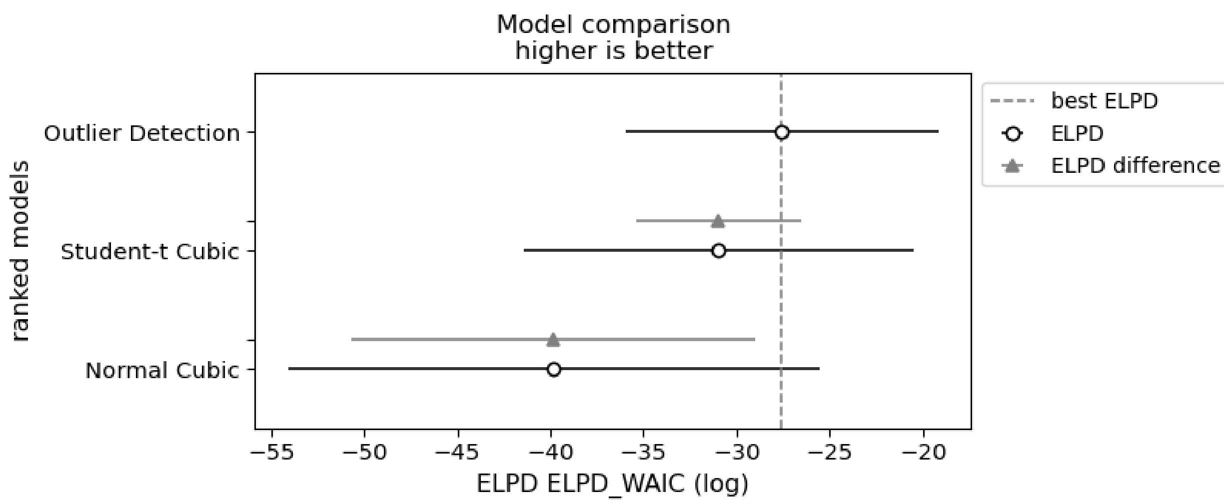
Now, when comparing the best two normal models (quadratic and cubic) with the best two student-t models (quadratic and cubic), the **student-t cubic one performed best with the ELPD_LOO and ELPD_WAICof -31**. Same as above, the error bars of the ELPD difference of the other 3 models relative to the best student-t cubic one does not overlap. So, we can be confident to conclude that the student-t cubic is the best model in this case.



These figures show the results of the normalized weights ranging between 0 and 1, scaling them based on their minimum and maximum values. Points with lower PSIS weights will be more opaque (not transparent), while those with higher weights will be more transparent. High weights on outliers suggest that these points have a significant influence on the model's predictions. We can see that the student-t models handle these outliers pretty well in that 1) the outliers have lower weights (darker points) and 2) their lines of best fit are not bent down or affected too much by those points, especially around the 2004 point clusters with the range of -1 and -0.5 on the x-axis.

3.3 Normal vs. Student's T vs. Outlier Detection





The outlier detection model performed better compared to the student-t cubic and the normal cubic models (ELPD_LOO of -29 compared to -31 and -39, respectively). However, unlikely the previous cases, the difference error bar of the student-t cubic overlaps with the best ELPD, which means that the credible interval of the student-t cubic's ELPD is likely to contain the same value as the outlier detection. In such as case, the difference between the two model's performance is 0. As a result, the outlier detection model and the student-t cubic model are relatively similar to each other. **Due to the complexity and computational expense that the outlier detection model entails, we might want to prefer the student-t cubic. However, given that we want to find a model that can best predict GDP based on government spending to inform better policy or expenditure strategy, I would still prefer and suggest the outlier detection given its out-of-sample performance**.

4 References

"How Government Spending Slows Growth." NBER, https://www.nber.org/digest/jan00/how-government-spending-slows-growth. Accessed 30 Oct. 2024.

Glossary | DataBank. https://databank.worldbank.org/metadataglossary/world-development-indicators/series/NE.CON.GOVT.ZS%23:~:text=Long%20definition,(including%20compensation%20of%20employees). Accessed 30 Oct. 2024.

International Monetary Fund, Real General Government Final Consumption Expenditure for Argentina [NCGGRNSAXDCARQ], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/NCGGRNSAXDCARQ, October 30, 2024.

International Monetary Fund, Real Gross Domestic Product for Argentina [NGDPRSAXDCARQ], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/NGDPRSAXDCARQ, October 30, 2024.

CS146 Session 12 - [7.1] Model comparison 3: Practice. Pre-class Work Notebook.

CS146 Session 8 - [4.2] Robust linear regression. Pre-class Work Notebook.

5 AI Statement

All the write-up is written by myself. There is no copy-paste interpretation from AI. I used Perplexity and Claude to debug some code related to visualizing the compare plot.

6 Code Appendix

6.1 Load Data

[128]:
```
import arviz as az
import numpy as np
import pandas as pd
import pymc as pm
import scipy.stats as sts
from scipy.special import logsumexp
import matplotlib.pyplot as plt
import seaborn as sns
```

[183]:
```
# Load dataset
gov_exp = pd.read_csv("Real General Government Final Consumption Expenditure for␣
 ↪Argentina.csv")
x = gov_exp["NCGGRSAXDCARQ"].values

gdp = pd.read_csv("Real Gross Domestic Product for Argentina.csv")
y = gdp["NGDPRSAXDCARQ"].values

# Create a dataframe
df = pd.DataFrame({"government_expenditure": x, "real_gdp": y})
df.head()
```

[183]:
```
   government_expenditure       real_gdp
0            13157.099609  118996.898438
1            13216.700195  117536.703125
2            13618.000000  123438.601562
3            13927.000000  125143.101562
4            14386.799805  128751.203125
```

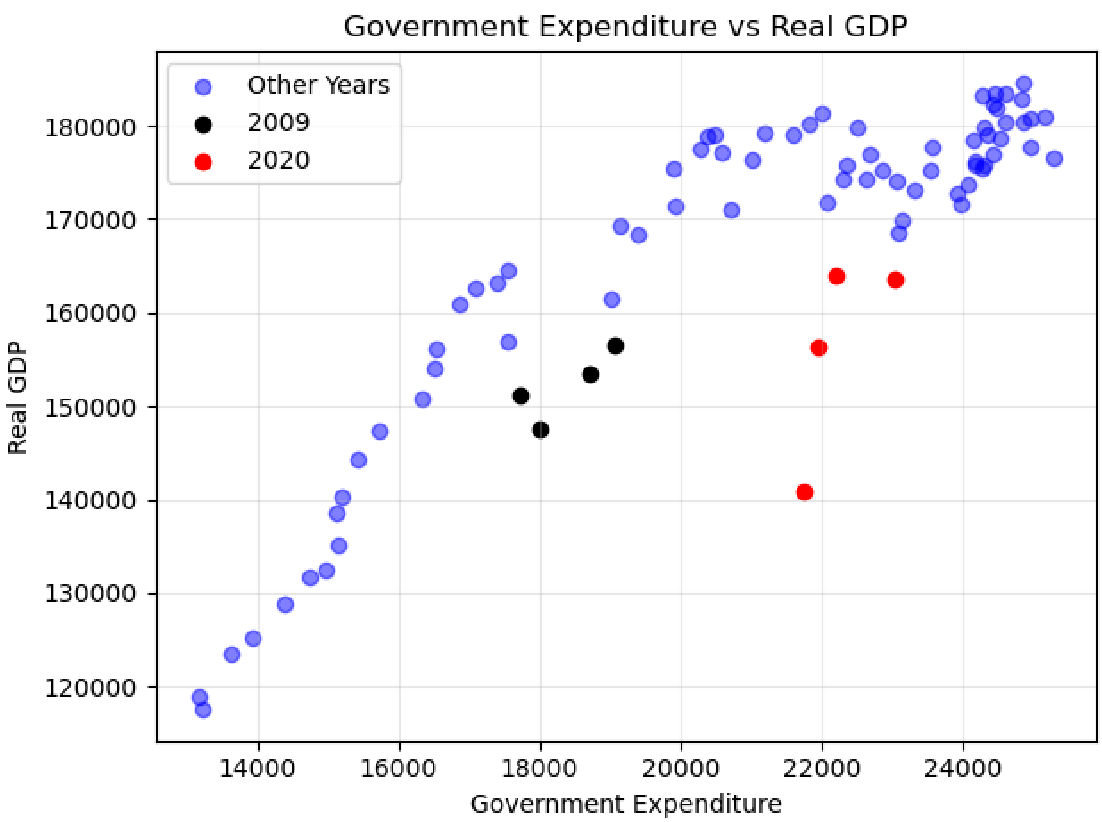
[130]:
```
date = gov_exp["DATE"]
df_date = pd.DataFrame({"date":date, "government_expenditure": x, "real_gdp": y})
df_date.head()
```

[130]:

	date	government_expenditure	real_gdp
0	2004-01-01	13157.099609	118996.898438
1	2004-04-01	13216.700195	117536.703125
2	2004-07-01	13618.000000	123438.601562
3	2004-10-01	13927.000000	125143.101562
4	2005-01-01	14386.799805	128751.203125

[131]:
```
def plot_gdp_expenditure(df):
    # Create a copy to avoid modifying the original dataframe
    df_plot = df_date.copy()

    # Convert date strings to datetime objects
    df_plot['date'] = pd.to_datetime(df_plot['date'])

    # Create boolean masks for 2019 and 2020 data
    mask_2020 = df_plot['date'].dt.year == 2020
    mask_2019 = df_plot['date'].dt.year == 2009
    mask_other = ~(mask_2019 | mask_2020)  # All other years

    # Plot other years data points
    plt.scatter(df_plot[mask_other]['government_expenditure'],
                df_plot[mask_other]['real_gdp'],
                color='blue',
                label='Other Years',
                alpha=0.5)

    # Plot 2019 data points
    if mask_2019.any():  # Only plot if there are 2019 points
        plt.scatter(df_plot[mask_2019]['government_expenditure'],
                    df_plot[mask_2019]['real_gdp'],
                    color='black',
                    label='2009')

    # Plot 2020 data points
    if mask_2020.any():  # Only plot if there are 2020 points
        plt.scatter(df_plot[mask_2020]['government_expenditure'],
```

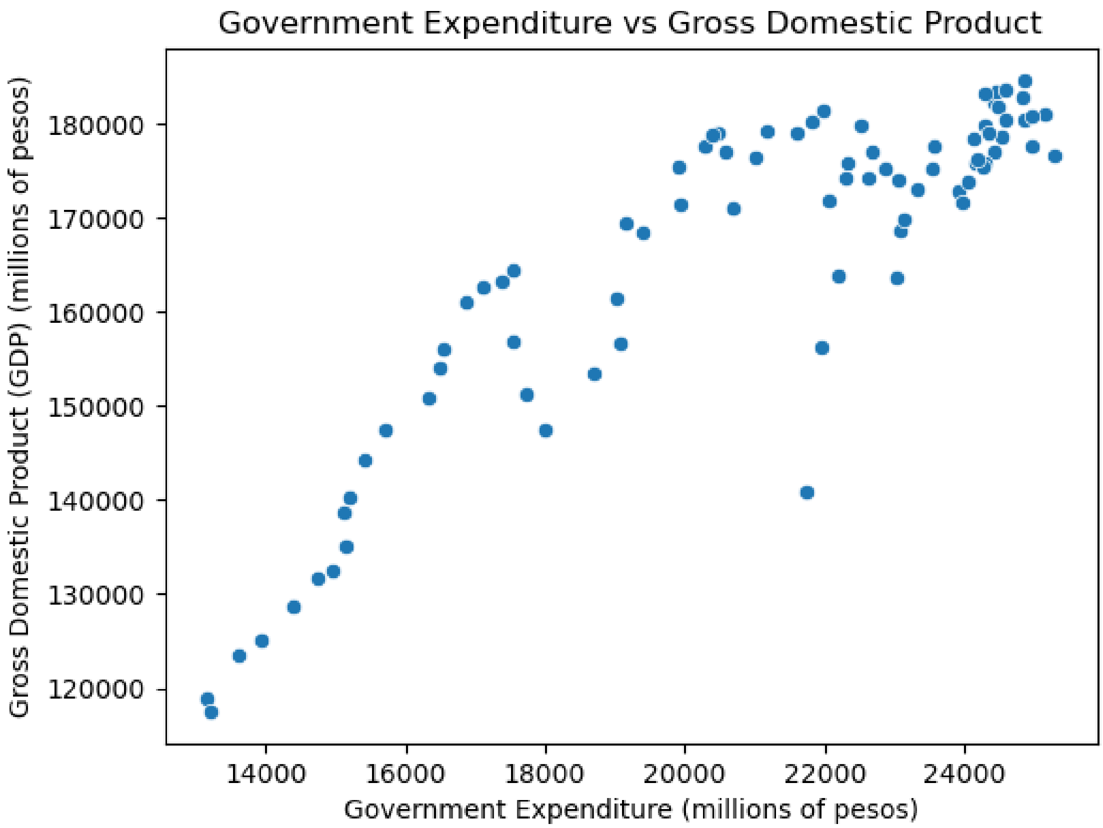
```
                         df_plot[mask_2020]['real_gdp'],
                         color='red',
                         label='2020')

    # Customize the plot
    plt.xlabel('Government Expenditure')
    plt.ylabel('Real GDP')
    plt.title('Government Expenditure vs Real GDP')
    plt.grid(True, alpha=0.3)
    plt.legend()

    # Show the plot
    plt.tight_layout()
    plt.show()

# Example usage with your dataframe:
plot_gdp_expenditure(df)
```



[132]:
```
sns.scatterplot(x=x, y=y)
plt.xlabel("Government Expenditure (millions of pesos)")
```

```
plt.ylabel("Gross Domestic Product (GDP) (millions of pesos)")
plt.title("Government Expenditure vs Gross Domestic Product")
```

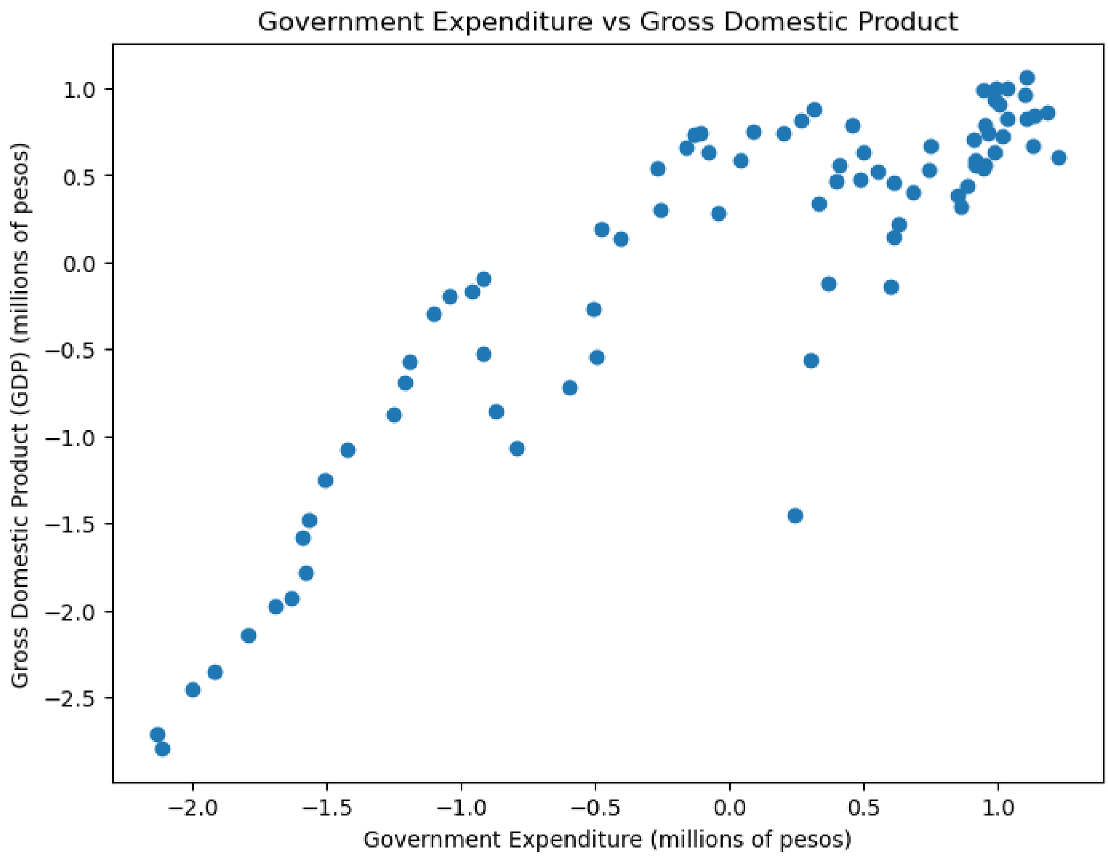
[132]: Text(0.5, 1.0, 'Government Expenditure vs Gross Domestic Product')



[184]:
```
df['gov_exp_std'] = (df['government_expenditure'] - df['government_expenditure'].
↪mean()) / df['government_expenditure'].std()
df['gdp_std'] = (df['real_gdp'] - df['real_gdp'].mean()) / df['real_gdp'].std()
x = df['gov_exp_std'].values
y = df['gdp_std'].values
```

[134]:
```
plt.figure(figsize=(8, 6))
plt.scatter(x, y)
plt.xlabel("Government Expenditure (millions of pesos)")
plt.ylabel("Gross Domestic Product (GDP) (millions of pesos)")
plt.title("Government Expenditure vs Gross Domestic Product")
```

[134]: Text(0.5, 1.0, 'Government Expenditure vs Gross Domestic Product')



[135]:
```
data_x = (df['government_expenditure'] - df['government_expenditure'].mean()) /␣
↪df['government_expenditure'].std()
data_y = (df['real_gdp'] - df['real_gdp'].mean()) / df['real_gdp'].std()
index = np.argsort(data_x)
data_x = data_x[index]
data_y = data_y[index]
```

6.2 Normal Model

[136]:
```
print('Fitting linear model')
with pm.Model() as model_1:
    a = pm.Uniform('a', lower=0, upper=20)
    b = pm.Normal('b', mu=0, sigma=1)
    sigma = pm.HalfNormal('sigma', 5)
    mu = pm.Deterministic('mu', a + b * data_x)
    y = pm.Normal('y', mu=mu, sigma=sigma, observed=data_y)
    inference_1 = pm.sample()
    pm.compute_log_likelihood(inference_1)
az.summary(inference_1, var_names="~mu")
```

```
Fitting linear model

Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [a, b, sigma]

Output()
```

```
Sampling 4 chains for 1_000 tune and 1_000 draw iterations (4_000 + 4_000 draws
total) took 1 seconds.
There were 5 divergences after tuning. Increase `target_accept` or
reparameterize.

Output()
```

[136]:

```
        mean     sd  hdi_5.5%  hdi_94.5%  mcse_mean  mcse_sd  ess_bulk  \
a      0.044  0.032     0.000      0.089      0.001    0.000    1695.0
b      0.877  0.054     0.793      0.963      0.001    0.001    1979.0
sigma  0.486  0.038     0.423      0.543      0.001    0.001    2167.0

       ess_tail  r_hat
a        1227.0    1.0
b        2028.0    1.0
sigma    2365.0    1.0
```

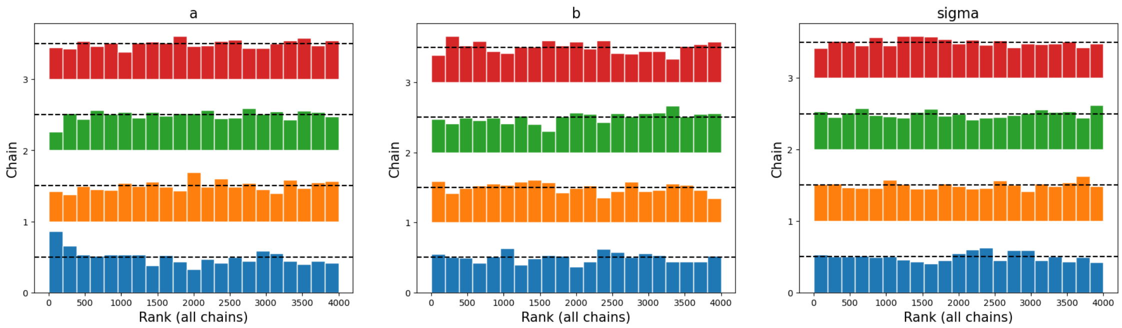
6.2.1 Visual Diagnostics

[137]:

```
az.plot_rank(inference_1, var_names=['a', 'b', 'sigma'])
```

[137]:

```
array([<Axes: title={'center': 'a'}, xlabel='Rank (all chains)',
ylabel='Chain'>,
       <Axes: title={'center': 'b'}, xlabel='Rank (all chains)',
ylabel='Chain'>,
       <Axes: title={'center': 'sigma'}, xlabel='Rank (all chains)',
ylabel='Chain'>],
      dtype=object)
```



[138]:
```
with model_1:
    gdp_pred_1 = pm.sample_posterior_predictive(inference_1)
```

```
Sampling: [y]
```

```
Output()
```

[139]:
```
print('Fitting quadratic model')
with pm.Model() as model_2:
    a = pm.Uniform('a', lower=0, upper=20)
    b = pm.Normal('b', mu=0, sigma=1)
    c = pm.Normal('c', mu=0, sigma=1)
    sigma = pm.HalfNormal('sigma', 5)
    mu = pm.Deterministic('mu', a + b * data_x + c * data_x**2)
    y = pm.Normal('y', mu=mu, sigma=sigma, observed=data_y)
    inference_2 = pm.sample(10000)
    pm.compute_log_likelihood(inference_2)
az.summary(inference_2, var_names="~mu")
```

```
Fitting quadratic model
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [a, b, c, sigma]
```

```
Output()
```

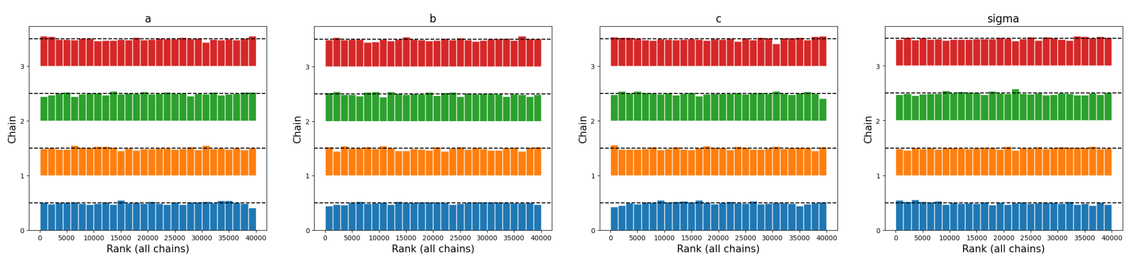
```
Sampling 4 chains for 1_000 tune and 10_000 draw iterations (4_000 + 40_000
draws total) took 5 seconds.
```

```
Output()
```

[139]:
```
        mean     sd  hdi_5.5%  hdi_94.5%  mcse_mean  mcse_sd  ess_bulk  \
a      0.333  0.068     0.224      0.442        0.0      0.0   20062.0
b      0.666  0.056     0.579      0.756        0.0      0.0   22438.0
c     -0.337  0.053    -0.422     -0.253        0.0      0.0   19431.0
sigma  0.397  0.032     0.345      0.447        0.0      0.0   27876.0

       ess_tail  r_hat
a       20017.0    1.0
b       23549.0    1.0
c       23119.0    1.0
sigma   25870.0    1.0
```

[140]:
```python
az.plot_rank(inference_2, var_names=['a', 'b', 'c', 'sigma'])
```

[140]:
```
array([<Axes: title={'center': 'a'}, xlabel='Rank (all chains)',
ylabel='Chain'>,
       <Axes: title={'center': 'b'}, xlabel='Rank (all chains)',
ylabel='Chain'>,
       <Axes: title={'center': 'c'}, xlabel='Rank (all chains)',
ylabel='Chain'>,
       <Axes: title={'center': 'sigma'}, xlabel='Rank (all chains)',
ylabel='Chain'>],
      dtype=object)
```



[141]:
```python
with model_2:
    gdp_pred_2 = pm.sample_posterior_predictive(inference_2)
```

```
Sampling: [y]

Output()
```

[142]:
```
print('Fitting cubic model')
with pm.Model() as model_3:
    a = pm.Uniform('a', lower=0, upper=20)
    b = pm.Normal('b', mu=0, sigma=1)
    c = pm.Normal('c', mu=0, sigma=1)
    d = pm.Normal('d', mu=0, sigma=1)
    sigma = pm.HalfNormal('sigma', 5)
    mu = pm.Deterministic('mu', a + b * data_x + c * data_x**2 + d * data_x**3)
    y = pm.Normal('y', mu=mu, sigma=sigma, observed=data_y)
    inference_3 = pm.sample(10000)
    pm.compute_log_likelihood(inference_3)
az.summary(inference_3, var_names="~mu")
```

```
Fitting cubic model

Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [a, b, c, d, sigma]

Output()

Sampling 4 chains for 1_000 tune and 10_000 draw iterations (4_000 + 40_000
draws total) took 9 seconds.

Output()
```

[142]:

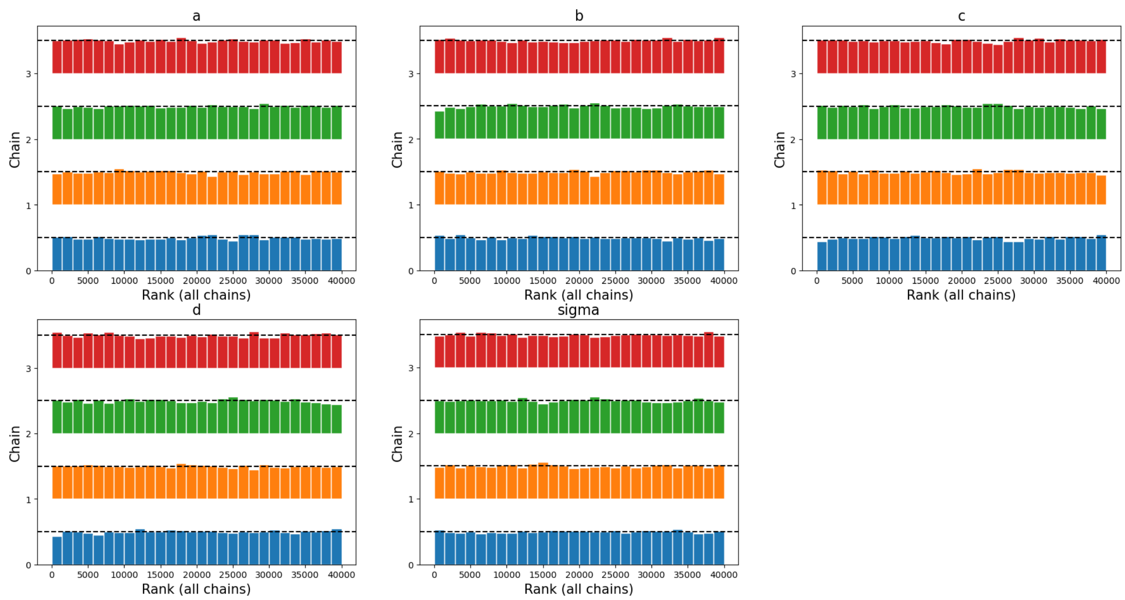
	mean	sd	hdi_5.5%	hdi_94.5%	mcse_mean	mcse_sd	ess_bulk	\
a	0.252	0.072	0.139	0.367	0.001	0.0	20179.0	
b	0.444	0.095	0.294	0.598	0.001	0.0	21864.0	
c	-0.152	0.083	-0.286	-0.023	0.001	0.0	17510.0	
d	0.164	0.058	0.070	0.255	0.000	0.0	17747.0	
sigma	0.380	0.031	0.329	0.428	0.000	0.0	24818.0	

	ess_tail	r_hat
a	15561.0	1.0
b	21411.0	1.0
c	18513.0	1.0
d	20655.0	1.0

```
sigma   23114.0    1.0
```

[143]:
```
az.plot_rank(inference_3, var_names=['a', 'b', 'c', 'd', 'sigma'])
```

[143]:
```
array([[<Axes: title={'center': 'a'}, xlabel='Rank (all chains)',
ylabel='Chain'>,
         <Axes: title={'center': 'b'}, xlabel='Rank (all chains)',
ylabel='Chain'>,
         <Axes: title={'center': 'c'}, xlabel='Rank (all chains)',
ylabel='Chain'>],
        [<Axes: title={'center': 'd'}, xlabel='Rank (all chains)',
ylabel='Chain'>,
         <Axes: title={'center': 'sigma'}, xlabel='Rank (all chains)',
ylabel='Chain'>,
         <Axes: >]], dtype=object)
```



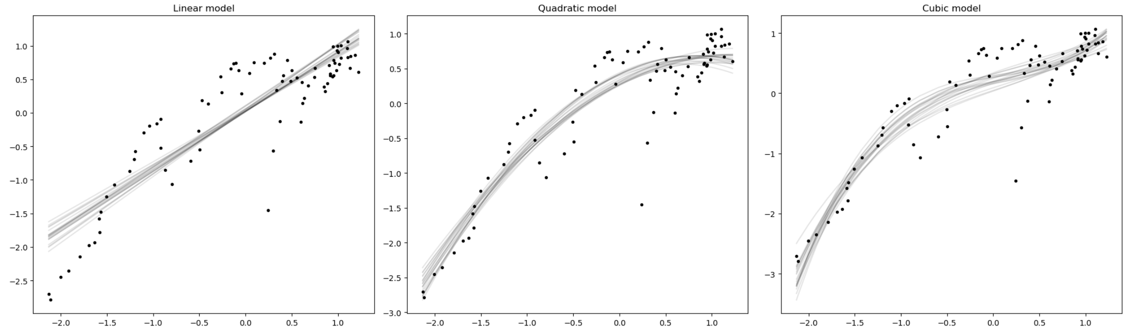
[144]:
```
with model_3:
    gdp_pred_3 = pm.sample_posterior_predictive(inference_3)
```

```
Sampling: [y]
```

```
Output()
```

6.2.2 Posterior Distribution

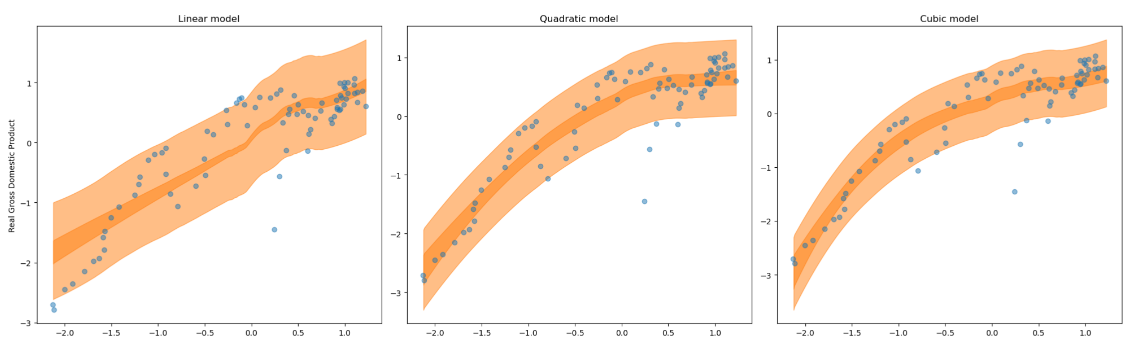
[145]:
```
plt.figure(figsize=(20, 6))

# Linear model
plt.subplot(1, 3, 1)
plt.title('Linear model')
plt.plot(data_x, data_y, 'k.')
plt.plot(data_x, inference_1.posterior.mu[0, :20, :].transpose(), 'k-', alpha=0.
↪1)

# Quadratic model
plt.subplot(1, 3, 2)
plt.title('Quadratic model')
plt.plot(data_x, data_y, 'k.')
plt.plot(data_x, inference_2.posterior.mu[0, :20, :].transpose(), 'k-', alpha=0.
↪1)

# Cubic model
plt.subplot(1, 3, 3)
plt.title('Cubic model')
plt.plot(data_x, data_y, 'k.')
plt.plot(data_x, inference_3.posterior.mu[0, :20, :].transpose(), 'k-', alpha=0.
↪1)
plt.tight_layout()
plt.show()
```



6.2.3 Posterior-predictive Distribution

[147]:
```
plt.figure(figsize=(20, 6))
az.rcParams["stats.ci_prob"] = 0.89

# Linear model
plt.subplot(1, 3, 1)
plt.title('Linear model')
```

```python
ax1 = az.plot_hdi(df['gov_exp_std'], inference_1.posterior["mu"])
az.plot_hdi(df['gov_exp_std'], gdp_pred_1.posterior_predictive["y"])
plt.scatter(df['gov_exp_std'], df['gdp_std'], alpha=0.5)
plt.ylabel("Real Gross Domestic Product")

# Quadratic model
plt.subplot(1, 3, 2)
plt.title('Quadratic model')
ax2 = az.plot_hdi(df['gov_exp_std'], inference_2.posterior["mu"])
az.plot_hdi(df['gov_exp_std'], gdp_pred_2.posterior_predictive["y"])
plt.scatter(df['gov_exp_std'], df['gdp_std'], alpha=0.5)

# Cubic model
plt.subplot(1, 3, 3)
plt.title('Cubic model')
ax3 = az.plot_hdi(df['gov_exp_std'], inference_3.posterior["mu"])
az.plot_hdi(df['gov_exp_std'], gdp_pred_3.posterior_predictive["y"])
plt.scatter(df['gov_exp_std'], df['gdp_std'], alpha=0.5)
plt.tight_layout()
plt.show()
```



6.2.4 Model Comparison (PSIS & WAIC)

[148]:
```python
print('Model comparison with PSIS')
df = az.compare({'linear': inference_1, 'quadratic': inference_2, 'cubic':␣
↪inference_3}, ic='loo')
df
```

```
Model comparison with PSIS
```

```
/Users/thananhthu/anaconda3/lib/python3.11/site-
packages/arviz/stats/stats.py:792: UserWarning: Estimated shape parameter of
Pareto distribution is greater than 0.70 for one or more samples. You should
consider using a more robust model, this is because importance sampling is less
likely to work well if the marginal posterior and LOO posterior are very
```

```
different. This is more likely to happen with a non-robust model and highly
influential observations.
  warnings.warn(
```

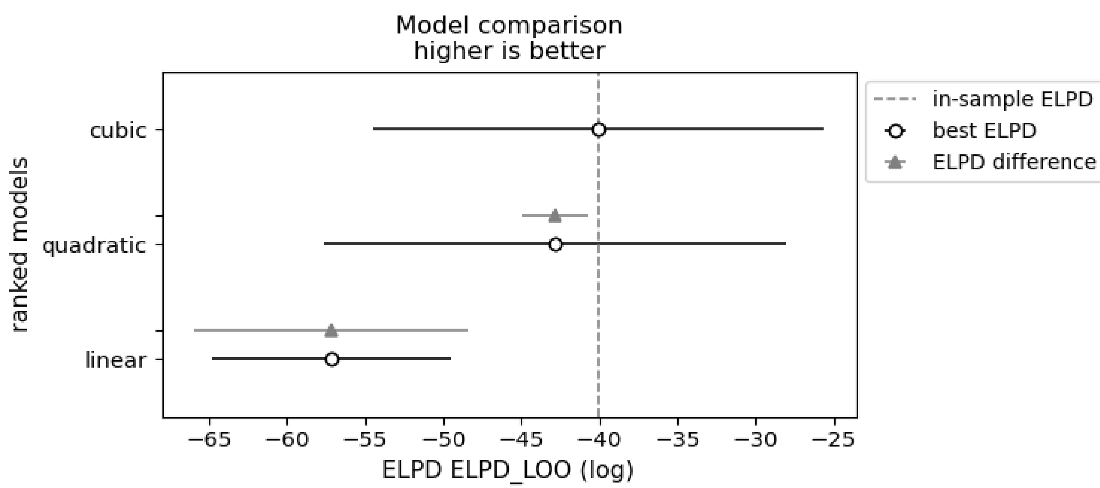
[148]:

	rank	elpd_loo	p_loo	elpd_diff	weight	se	\
cubic	0	-40.041858	7.268029	0.000000	9.225708e-01	14.458054	
quadratic	1	-42.809987	6.332804	2.768129	4.230556e-16	14.787892	
linear	2	-57.177354	2.686335	17.135496	7.742920e-02	7.645223	

	dse	warning	scale
cubic	0.000000	True	log
quadratic	2.109632	False	log
linear	8.803936	False	log

[149]:

```
az.plot_compare(df)
plt.title('Model comparison\nhigher is better')
plt.xlabel('ELPD ' + df.columns[1].upper() + ' (log)')
plt.legend(['in-sample ELPD', 'best ELPD', 'ELPD difference', 'ELPD'],␣
 ↪loc='upper left', bbox_to_anchor=(1.0, 1.0))
plt.show()
```



[150]:

```
print('Model comparison with WAIC')
df = az.compare({'linear': inference_1, 'quadratic': inference_2, 'cubic':␣
 ↪inference_3}, ic='waic')
df
```

```
Model comparison with WAIC
```

```
/Users/thananhthu/anaconda3/lib/python3.11/site-
packages/arviz/stats/stats.py:1647: UserWarning: For one or more samples the
posterior variance of the log predictive densities exceeds 0.4. This could be
```

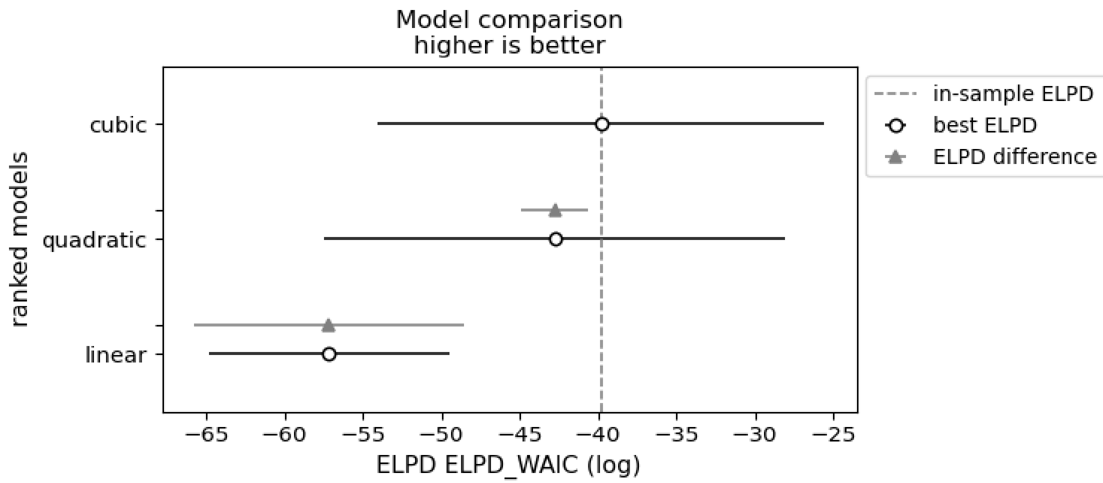
```
indication of WAIC starting to fail.
See http://arxiv.org/abs/1507.04544 for details
  warnings.warn(
/Users/thananhthu/anaconda3/lib/python3.11/site-
packages/arviz/stats/stats.py:1647: UserWarning: For one or more samples the
posterior variance of the log predictive densities exceeds 0.4. This could be
indication of WAIC starting to fail.
See http://arxiv.org/abs/1507.04544 for details
  warnings.warn(
/Users/thananhthu/anaconda3/lib/python3.11/site-
packages/arviz/stats/stats.py:1647: UserWarning: For one or more samples the
posterior variance of the log predictive densities exceeds 0.4. This could be
indication of WAIC starting to fail.
See http://arxiv.org/abs/1507.04544 for details
  warnings.warn(
```

[150]:

```
           rank  elpd_waic    p_waic  elpd_diff        weight         se  \
cubic         0 -39.810245  7.036415   0.000000  9.228412e-01  14.263484
quadratic     1 -42.768533  6.291350   2.958288  4.206034e-16  14.754359
linear        2 -57.170879  2.679861  17.360635  7.715884e-02   7.647108

                dse  warning scale
cubic      0.000000     True   log
quadratic  2.140236     True   log
linear     8.628996     True   log
```

[151]:

```
az.plot_compare(df)
# Add these labels since they are not in the latest release (scheduled for the␣
↪next one)
plt.title('Model comparison\nhigher is better')
plt.xlabel('ELPD ' + df.columns[1].upper() + ' (log)')
plt.legend(['in-sample ELPD', 'best ELPD', 'ELPD difference', 'ELPD'],␣
↪loc='upper left', bbox_to_anchor=(1.0, 1.0))
plt.show()
```



6.3 Student-T Likelihood

[152]:
```
print('Fitting linear model')
with pm.Model() as model_1:
    a = pm.Uniform('a', lower=0, upper=20)
    b = pm.Normal('b', mu=0, sigma=1)
    sigma = pm.HalfNormal('sigma', 5)
    nu = pm.HalfNormal('nu', 30)
    mu = pm.Deterministic('mu', a + b * data_x)
    y = pm.StudentT('y', nu=nu, mu=mu, sigma=sigma, observed=data_y)
    inference_1t = pm.sample(tune=2000)
    pm.compute_log_likelihood(inference_1t)
az.summary(inference_1t, var_names="~mu")
```

```
Fitting linear model
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [a, b, sigma, nu]
```

```
Output()
```

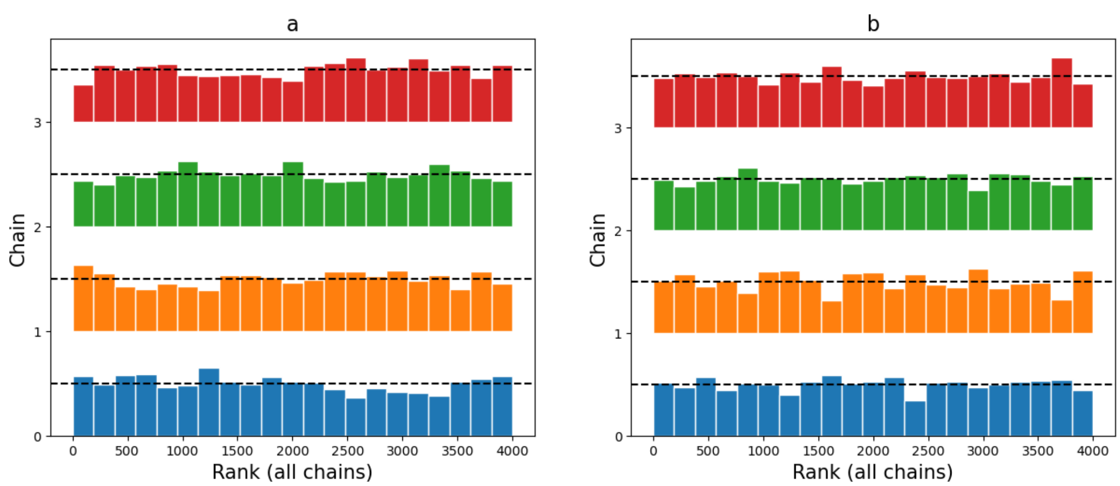
```
Sampling 4 chains for 2_000 tune and 1_000 draw iterations (8_000 + 4_000 draws
total) took 2 seconds.
There were 2 divergences after tuning. Increase `target_accept` or
reparameterize.
```

```
Output()
```

```
[152]:          mean      sd  hdi_5.5%  hdi_94.5%  mcse_mean  mcse_sd  ess_bulk  \
       a       0.045   0.034     0.000      0.088      0.001    0.000    1661.0
       b       0.869   0.054     0.780      0.952      0.001    0.001    2518.0
       nu     26.439  16.210     4.557     48.977      0.325    0.230    2160.0
       sigma   0.460   0.043     0.390      0.525      0.001    0.001    2468.0

              ess_tail  r_hat
       a        1040.0    1.0
       b        2313.0    1.0
       nu       1873.0    1.0
       sigma    2230.0    1.0
```

```
[153]: az.plot_rank(inference_1t, var_names=['a', 'b'])
```

```
[153]: array([<Axes: title={'center': 'a'}, xlabel='Rank (all chains)',
       ylabel='Chain'>,
              <Axes: title={'center': 'b'}, xlabel='Rank (all chains)',
       ylabel='Chain'>],
             dtype=object)
```




```
[154]: print('Fitting quadratic model')
       with pm.Model() as model_2:
           a = pm.Uniform('a', lower=0, upper=20)
           b = pm.Normal('b', mu=0, sigma=1)
           c = pm.Normal('c', mu=0, sigma=1)
           sigma = pm.HalfNormal('sigma', 5)
```

```
    nu = pm.HalfNormal('nu', 30)
    mu = pm.Deterministic('mu', a + b * data_x + c * data_x**2)
    y = pm.StudentT('y', nu=nu, mu=mu, sigma=sigma, observed=data_y)
    inference_2t = pm.sample(10000, tune=2000)
    pm.compute_log_likelihood(inference_2t)
az.summary(inference_2t, var_names="~mu")
```

```
Fitting quadratic model
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [a, b, c, sigma, nu]
```

```
Output()
```

```
Sampling 4 chains for 2_000 tune and 10_000 draw iterations (8_000 + 40_000
draws total) took 7 seconds.
```

```
Output()
```

[154]:

```
        mean     sd  hdi_5.5%  hdi_94.5%  mcse_mean  mcse_sd  ess_bulk  \
a      0.440  0.067     0.334      0.546      0.001    0.000   14522.0
b      0.643  0.046     0.571      0.716      0.000    0.000   18463.0
c     -0.393  0.045    -0.468     -0.324      0.000    0.000   14533.0
nu     6.104  4.610     1.812     10.170      0.034    0.024   20282.0
sigma  0.282  0.039     0.220      0.343      0.000    0.000   18356.0

       ess_tail  r_hat
a       18280.0    1.0
b       24154.0    1.0
c       18645.0    1.0
nu      21909.0    1.0
sigma   19256.0    1.0
```

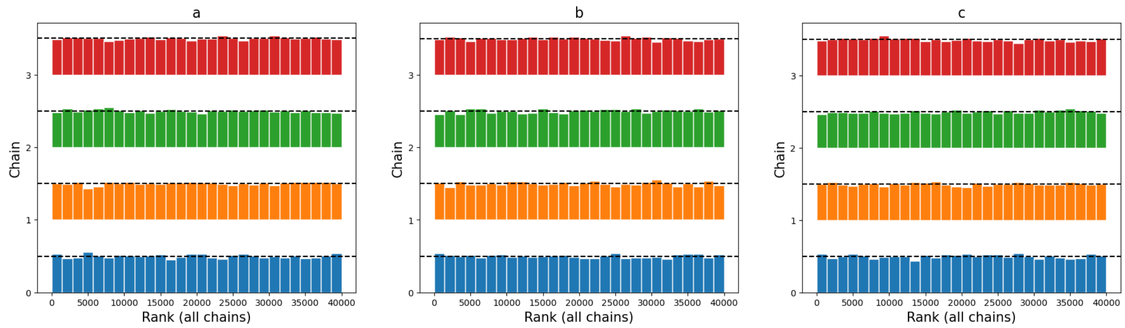
[155]:

```
az.plot_rank(inference_2t, var_names=['a', 'b', 'c'])
```

[155]:

```
array([<Axes: title={'center': 'a'}, xlabel='Rank (all chains)',
ylabel='Chain'>,
       <Axes: title={'center': 'b'}, xlabel='Rank (all chains)',
ylabel='Chain'>,
       <Axes: title={'center': 'c'}, xlabel='Rank (all chains)',
```

```
ylabel='Chain'>],
      dtype=object)
```



[156]:

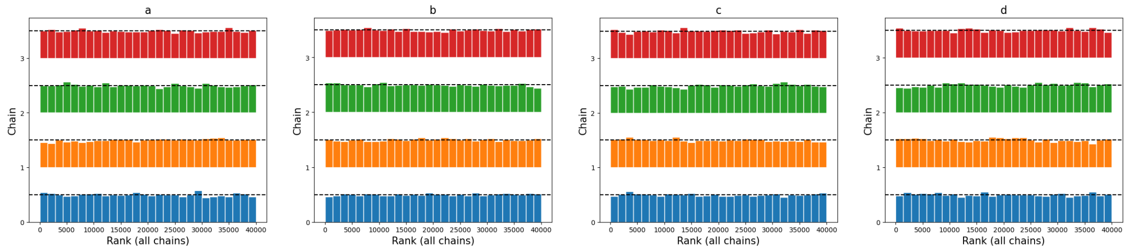
```python
print('Fitting cubic model')
with pm.Model() as model_3:
    a = pm.Uniform('a', lower=0, upper=20)
    b = pm.Normal('b', mu=0, sigma=1)
    c = pm.Normal('c', mu=0, sigma=1)
    d = pm.Normal('d', mu=0, sigma=1)
    sigma = pm.HalfNormal('sigma', 5)
    nu = pm.HalfNormal('nu', 30)
    mu = pm.Deterministic('mu', a + b * data_x + c * data_x**2 + d * data_x**3)
    y = pm.StudentT('y', nu=nu, mu=mu, sigma=sigma, observed=data_y)
    inference_3t = pm.sample(10000, tune=2000)
    pm.compute_log_likelihood(inference_3t)
az.summary(inference_3t, var_names="~mu")
```

```
Fitting cubic model

Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [a, b, c, d, sigma, nu]

Output()

Sampling 4 chains for 2_000 tune and 10_000 draw iterations (8_000 + 40_000
draws total) took 13 seconds.

Output()
```

[156]:
```
        mean     sd  hdi_5.5%  hdi_94.5%  mcse_mean  mcse_sd  ess_bulk  \
a      0.379  0.070     0.269      0.491      0.001    0.000   15646.0
b      0.449  0.080     0.321      0.576      0.001    0.000   19567.0
c     -0.245  0.070    -0.355     -0.132      0.001    0.000   14150.0
d      0.134  0.048     0.058      0.210      0.000    0.000   15420.0
nu     5.510  4.327     1.672      9.123      0.036    0.025   18112.0
sigma  0.263  0.039     0.201      0.324      0.000    0.000   17823.0

       ess_tail  r_hat
a       16772.0    1.0
b       23526.0    1.0
c       17422.0    1.0
d       19811.0    1.0
nu      18019.0    1.0
sigma   19884.0    1.0
```

[157]:
```python
az.plot_rank(inference_3t, var_names=['a', 'b', 'c', 'd'])
```

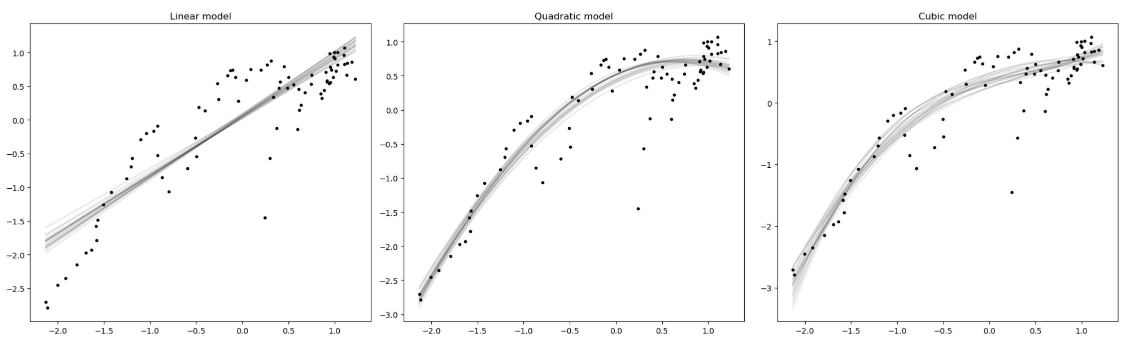
[157]:
```
array([<Axes: title={'center': 'a'}, xlabel='Rank (all chains)',
ylabel='Chain'>,
       <Axes: title={'center': 'b'}, xlabel='Rank (all chains)',
ylabel='Chain'>,
       <Axes: title={'center': 'c'}, xlabel='Rank (all chains)',
ylabel='Chain'>,
       <Axes: title={'center': 'd'}, xlabel='Rank (all chains)',
ylabel='Chain'>],
      dtype=object)
```



6.3.1 Posterior Distribution

[158]:
```python
plt.figure(figsize=(20, 6))

# Linear model
plt.subplot(1, 3, 1)
plt.title('Linear model')
```

```
plt.plot(data_x, data_y, 'k.')
plt.plot(data_x, inference_1t.posterior.mu[0, :20, :].transpose(), 'k-', alpha=0.
 ↪1)

# Quadratic model
plt.subplot(1, 3, 2)
plt.title('Quadratic model')
plt.plot(data_x, data_y, 'k.')
plt.plot(data_x, inference_2t.posterior.mu[0, :20, :].transpose(), 'k-', alpha=0.
 ↪1)

# Cubic model
plt.subplot(1, 3, 3)
plt.title('Cubic model')
plt.plot(data_x, data_y, 'k.')
plt.plot(data_x, inference_3t.posterior.mu[0, :20, :].transpose(), 'k-', alpha=0.
 ↪1)
plt.tight_layout()
plt.show()
```



[159]:
```
with model_1:
    gdp_pred_1t = pm.sample_posterior_predictive(inference_1t)
```

Sampling: [y]

Output()

[160]:
```
with model_2:
    gdp_pred_2t = pm.sample_posterior_predictive(inference_2t)
```

Sampling: [y]

Output()

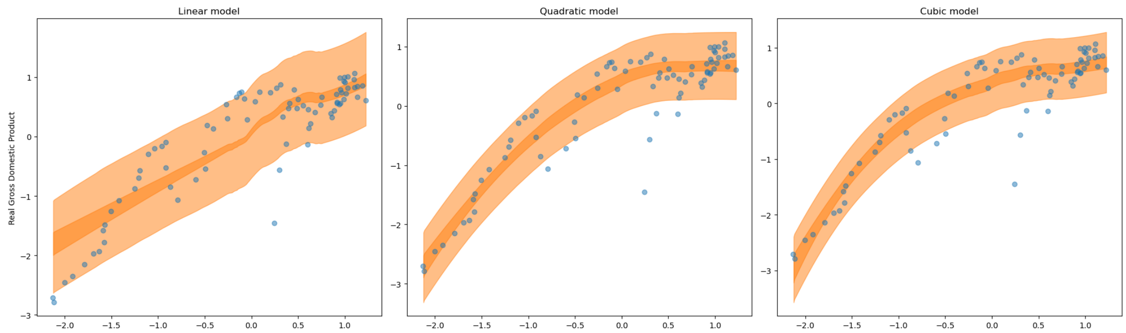
[161]:
```
with model_3:
    gdp_pred_3t = pm.sample_posterior_predictive(inference_3t)
```

Sampling: [y]

Output()

6.3.2 Posterior-predictive Distribution

[166]:
```
plt.figure(figsize=(20, 6))
az.rcParams["stats.ci_prob"] = 0.89

# Linear model
plt.subplot(1, 3, 1)
plt.title('Linear model')
ax1 = az.plot_hdi(df['gov_exp_std'], inference_1t.posterior["mu"])
az.plot_hdi(df['gov_exp_std'], gdp_pred_1t.posterior_predictive["y"])
plt.scatter(df['gov_exp_std'], df['gdp_std'], alpha=0.5)
plt.ylabel("Real Gross Domestic Product")

# Quadratic model
plt.subplot(1, 3, 2)
plt.title('Quadratic model')
ax2 = az.plot_hdi(df['gov_exp_std'], inference_2t.posterior["mu"])
az.plot_hdi(df['gov_exp_std'], gdp_pred_2t.posterior_predictive["y"])
plt.scatter(df['gov_exp_std'], df['gdp_std'], alpha=0.5)

# Cubic model
plt.subplot(1, 3, 3)
plt.title('Cubic model')
ax3 = az.plot_hdi(df['gov_exp_std'], inference_3t.posterior["mu"])
az.plot_hdi(df['gov_exp_std'], gdp_pred_3t.posterior_predictive["y"])
plt.scatter(df['gov_exp_std'], df['gdp_std'], alpha=0.5)
plt.tight_layout()
plt.show()
```



6.3.3 Model Comparison

[167]:
```
print('Model comparison with PSIS')
df = az.compare({'linear': inference_1t, 'quadratic': inference_2t, 'cubic':␣
↪inference_3t}, ic='loo')
df
```

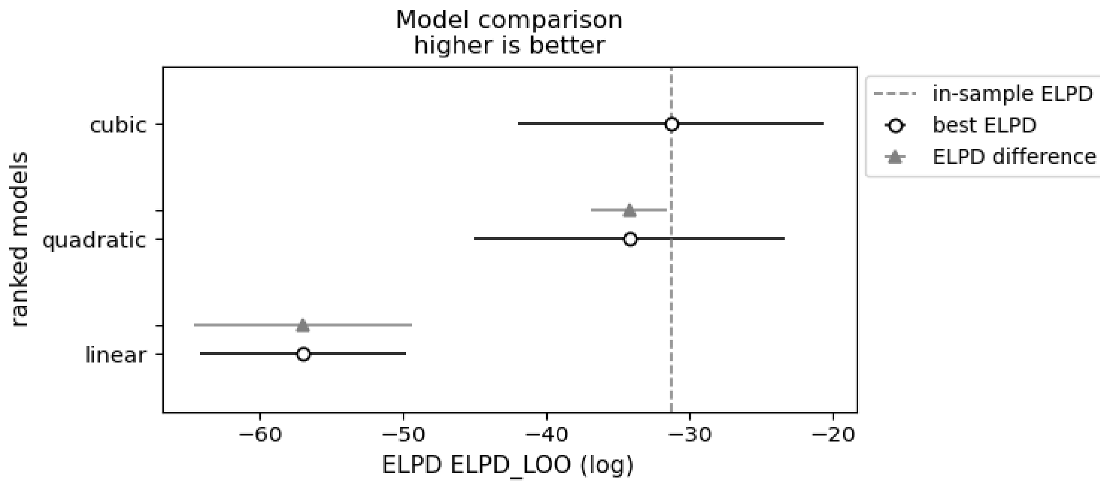
```
Model comparison with PSIS

/Users/thananhthu/anaconda3/lib/python3.11/site-
packages/arviz/stats/stats.py:792: UserWarning: Estimated shape parameter of
Pareto distribution is greater than 0.70 for one or more samples. You should
consider using a more robust model, this is because importance sampling is less
likely to work well if the marginal posterior and LOO posterior are very
different. This is more likely to happen with a non-robust model and highly
influential observations.
  warnings.warn(
```

[167]:

	rank	elpd_loo	p_loo	elpd_diff	weight	se	\
cubic	0	-31.299420	7.325974	0.000000	9.059501e-01	10.702758	
quadratic	1	-34.209595	6.103229	2.910175	7.204079e-18	10.821175	
linear	2	-57.027114	2.796452	25.727693	9.404988e-02	7.172433	

	dse	warning	scale
cubic	0.000000	False	log
quadratic	2.632483	True	log
linear	7.611378	False	log

[168]:
```
az.plot_compare(df)
plt.title('Model comparison\nhigher is better')
plt.xlabel('ELPD ' + df.columns[1].upper() + ' (log)')
plt.legend(['in-sample ELPD', 'best ELPD', 'ELPD difference', 'ELPD'],␣
↪loc='upper left', bbox_to_anchor=(1.0, 1.0))
plt.show()
```



[169]:
```
print('Model comparison with WAIC')
df = az.compare({'linear': inference_1t, 'quadratic': inference_2t, 'cubic':␣
↪inference_3t}, ic='waic')
df
```

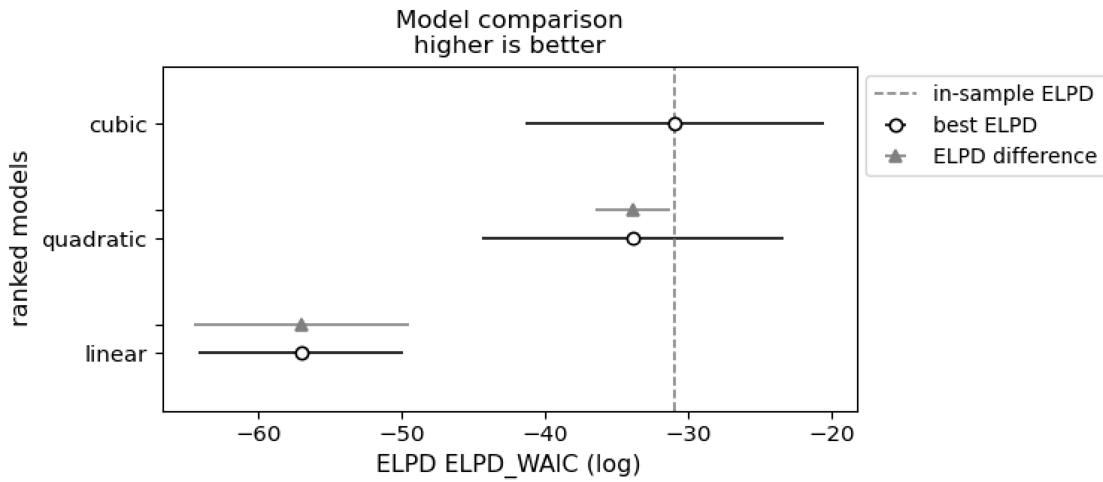
```
Model comparison with WAIC

/Users/thananhthu/anaconda3/lib/python3.11/site-
packages/arviz/stats/stats.py:1647: UserWarning: For one or more samples the
posterior variance of the log predictive densities exceeds 0.4. This could be
indication of WAIC starting to fail.
See http://arxiv.org/abs/1507.04544 for details
  warnings.warn(
/Users/thananhthu/anaconda3/lib/python3.11/site-
packages/arviz/stats/stats.py:1647: UserWarning: For one or more samples the
posterior variance of the log predictive densities exceeds 0.4. This could be
indication of WAIC starting to fail.
See http://arxiv.org/abs/1507.04544 for details
  warnings.warn(
/Users/thananhthu/anaconda3/lib/python3.11/site-
packages/arviz/stats/stats.py:1647: UserWarning: For one or more samples the
posterior variance of the log predictive densities exceeds 0.4. This could be
indication of WAIC starting to fail.
See http://arxiv.org/abs/1507.04544 for details
  warnings.warn(
```

[169]:

	rank	elpd_waic	p_waic	elpd_diff	weight	se	\
cubic	0	-30.884289	6.910842	0.000000	9.150001e-01	10.428327	
quadratic	1	-33.819702	5.713337	2.935414	2.805264e-17	10.531232	
linear	2	-57.023691	2.793029	26.139402	8.499990e-02	7.173391	

```
               dse  warning scale
cubic      0.000000     True   log
quadratic  2.626027     True   log
linear     7.498878     True   log
```

[170]:
```
az.plot_compare(df)
plt.title('Model comparison\nhigher is better')
plt.xlabel('ELPD ' + df.columns[1].upper() + ' (log)')
plt.legend(['in-sample ELPD', 'best ELPD', 'ELPD difference', 'ELPD'],␣
 ↪loc='upper left', bbox_to_anchor=(1.0, 1.0))
plt.show()
```



6.3.4 Model Comparison (Normal vs. Student's t)

[171]:
```
print('Model comparison with PSIS')
df = az.compare(
    {'Normal quadratic': inference_2,
     'Normal cubic': inference_3,
     'Student-T quadratic': inference_2t,
     'Student-T cubic': inference_3t
    },
    ic='loo')
df
```

```
Model comparison with PSIS
```

```
/Users/thananhthu/anaconda3/lib/python3.11/site-
packages/arviz/stats/stats.py:792: UserWarning: Estimated shape parameter of
Pareto distribution is greater than 0.70 for one or more samples. You should
consider using a more robust model, this is because importance sampling is less
likely to work well if the marginal posterior and LOO posterior are very
```

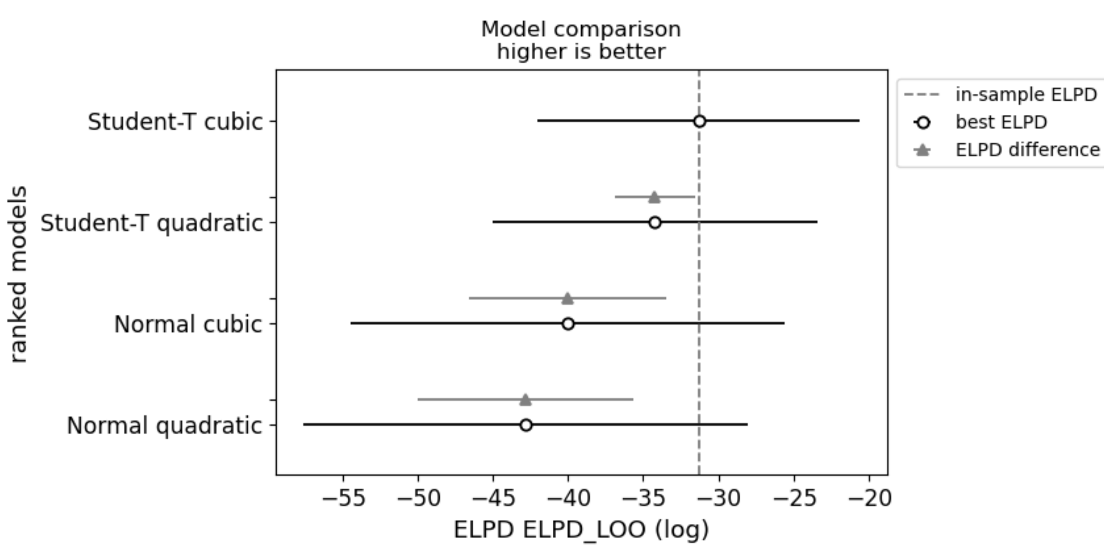
```
different. This is more likely to happen with a non-robust model and highly
influential observations.
  warnings.warn(
/Users/thananhthu/anaconda3/lib/python3.11/site-
packages/arviz/stats/stats.py:792: UserWarning: Estimated shape parameter of
Pareto distribution is greater than 0.70 for one or more samples. You should
consider using a more robust model, this is because importance sampling is less
likely to work well if the marginal posterior and LOO posterior are very
different. This is more likely to happen with a non-robust model and highly
influential observations.
  warnings.warn(
```

[171]:

```
                      rank   elpd_loo     p_loo  elpd_diff        weight  \
Student-T cubic          0 -31.299420  7.325974   0.000000  8.909511e-01
Student-T quadratic      1 -34.209595  6.103229   2.910175  1.629820e-17
Normal cubic             2 -40.041858  7.268029   8.742438  8.215026e-03
Normal quadratic         3 -42.809987  6.332804  11.510567  1.008339e-01

                            se       dse  warning scale
Student-T cubic      10.702758  0.000000    False   log
Student-T quadratic  10.821175  2.632483     True   log
Normal cubic         14.458054  6.559856     True   log
Normal quadratic     14.787892  7.199435    False   log
```

[172]:

```python
az.plot_compare(df)
plt.title('Model comparison\nhigher is better')
plt.xlabel('ELPD ' + df.columns[1].upper() + ' (log)')
plt.legend(['in-sample ELPD', 'best ELPD', 'ELPD difference', 'ELPD'],␣
 ↪loc='upper left', bbox_to_anchor=(1.0, 1.0))
plt.show()
```



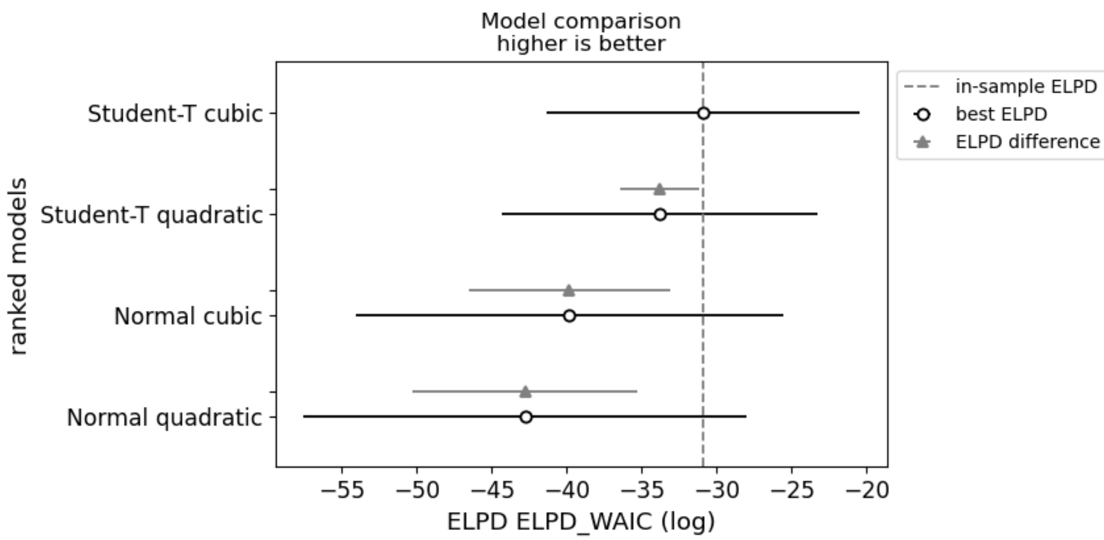
[173]:
```
print('Model comparison with WAIC')
df = az.compare(
    {'Normal quadratic': inference_2,
     'Normal cubic': inference_3,
     'Student-T quadratic': inference_2t,
     'Student-T cubic': inference_3t
    },
    ic='waic')
df
```

```
Model comparison with WAIC

/Users/thananhthu/anaconda3/lib/python3.11/site-
packages/arviz/stats/stats.py:1647: UserWarning: For one or more samples the
posterior variance of the log predictive densities exceeds 0.4. This could be
indication of WAIC starting to fail.
See http://arxiv.org/abs/1507.04544 for details
  warnings.warn(
/Users/thananhthu/anaconda3/lib/python3.11/site-
packages/arviz/stats/stats.py:1647: UserWarning: For one or more samples the
posterior variance of the log predictive densities exceeds 0.4. This could be
indication of WAIC starting to fail.
See http://arxiv.org/abs/1507.04544 for details
  warnings.warn(
/Users/thananhthu/anaconda3/lib/python3.11/site-
packages/arviz/stats/stats.py:1647: UserWarning: For one or more samples the
posterior variance of the log predictive densities exceeds 0.4. This could be
indication of WAIC starting to fail.
See http://arxiv.org/abs/1507.04544 for details
  warnings.warn(
/Users/thananhthu/anaconda3/lib/python3.11/site-
packages/arviz/stats/stats.py:1647: UserWarning: For one or more samples the
posterior variance of the log predictive densities exceeds 0.4. This could be
indication of WAIC starting to fail.
See http://arxiv.org/abs/1507.04544 for details
  warnings.warn(
```

[173]:
```
                     rank  elpd_waic    p_waic  elpd_diff        weight  \
Student-T cubic         0 -30.884289  6.910842   0.000000  8.911998e-01
Student-T quadratic     1 -33.819702  5.713337   2.935414  8.246866e-17
Normal cubic            2 -39.810245  7.036415   8.925956  1.021096e-02
Normal quadratic        3 -42.768533  6.291350  11.884244  9.858923e-02

                            se       dse  warning scale
Student-T cubic      10.428327  0.000000     True   log
Student-T quadratic  10.531232  2.626027     True   log
```

```
Normal cubic          14.263484  6.709990    True   log
Normal quadratic      14.754359  7.488449    True   log
```

[174]:
```
az.plot_compare(df)
plt.title('Model comparison\nhigher is better')
plt.xlabel('ELPD ' + df.columns[1].upper() + ' (log)')
plt.legend(['in-sample ELPD', 'best ELPD', 'ELPD difference', 'ELPD'],␣
 ↪loc='upper left', bbox_to_anchor=(1.0, 1.0))
plt.show()
```

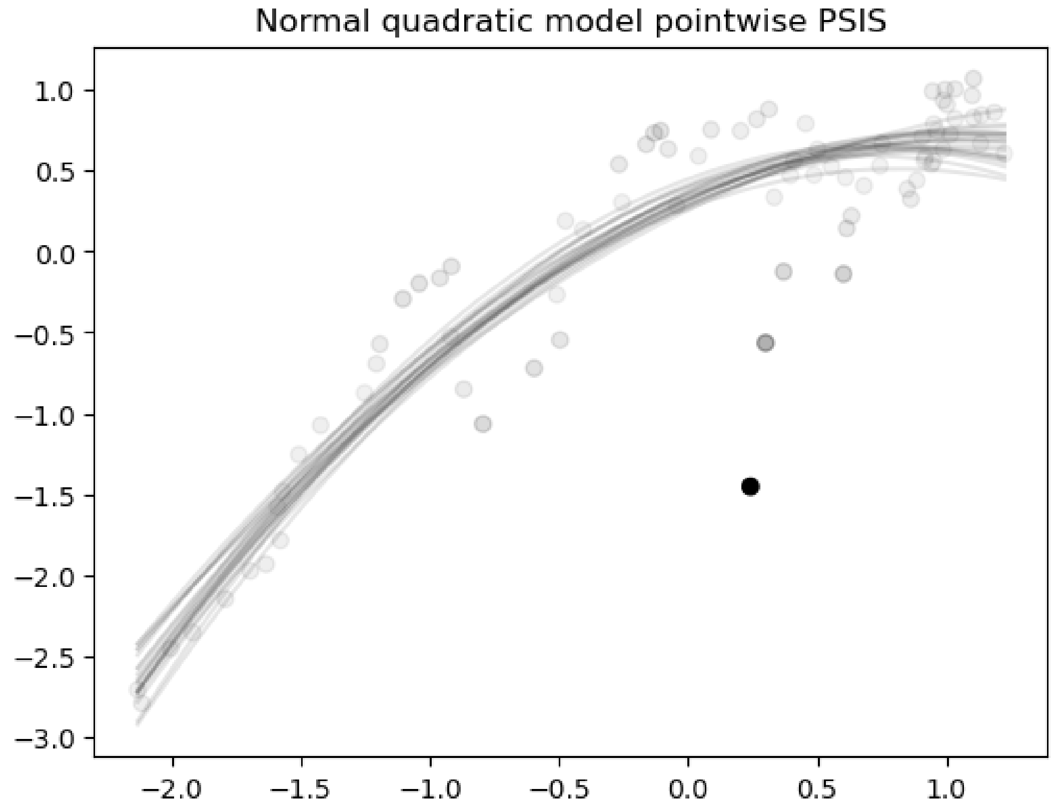


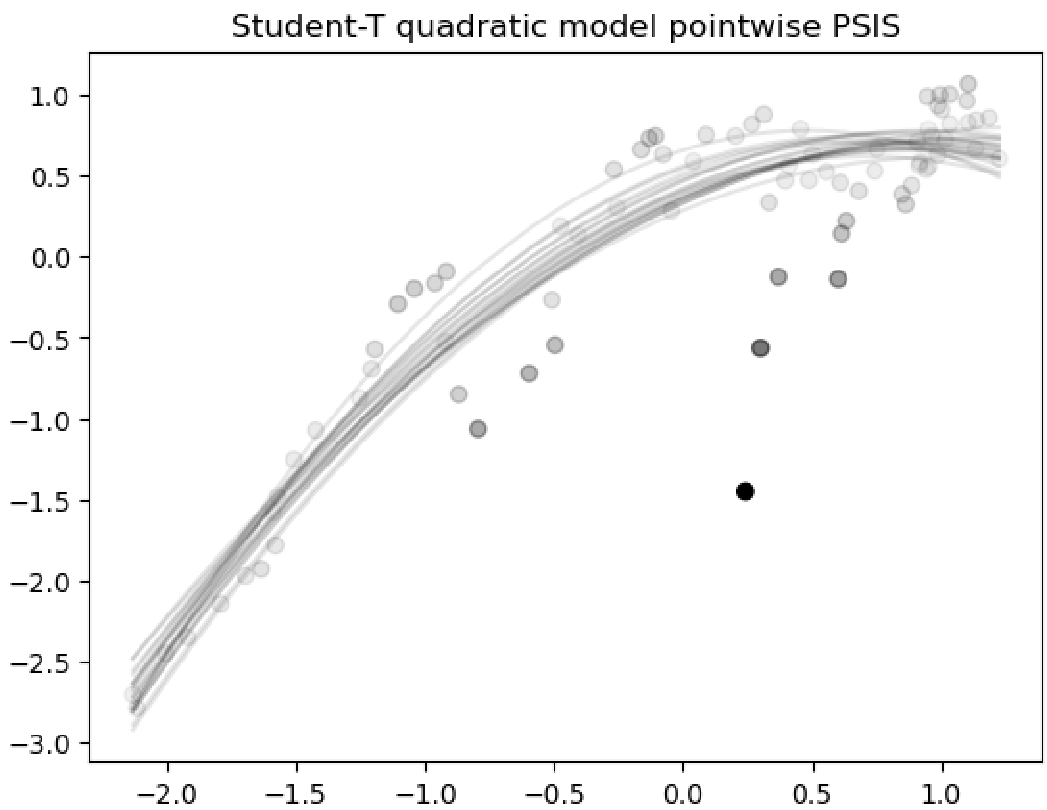
[46]:
```
plt.figure()
plt.title('Normal quadratic model pointwise PSIS')
weights = az.loo(inference_2, pointwise=True).loo_i.values
print('Pointwise PSIS:', weights)
alpha = 1 - 0.95 * (weights - weights.min()) / (weights.max() - weights.min())
plt.scatter(data_x, data_y, color='black', alpha=alpha)
plt.plot(data_x, inference_2.posterior.mu[0, :20, :].transpose(), 'k-', alpha=0.
 ↪1)

plt.figure()
plt.title('Student-T quadratic model pointwise PSIS')
weights = az.loo(inference_2t, pointwise=True).loo_i.values
print('Pointwise PSIS:', weights)
alpha = 1 - 0.95 * (weights - weights.min()) / (weights.max() - weights.min())
plt.scatter(data_x, data_y, color='black', alpha=alpha)
plt.plot(data_x, inference_2t.posterior.mu[0, :20, :].transpose(), 'k-', alpha=0.
 ↪1)
plt.show()
```

```
/Users/thananhthu/anaconda3/lib/python3.11/site-
packages/arviz/stats/stats.py:792: UserWarning: Estimated shape parameter of
Pareto distribution is greater than 0.70 for one or more samples. You should
consider using a more robust model, this is because importance sampling is less
likely to work well if the marginal posterior and LOO posterior are very
different. This is more likely to happen with a non-robust model and highly
influential observations.
  warnings.warn(
```

```
Pointwise PSIS: [-1.10724793e-01 -2.39706122e-01 -9.14832850e-02 -1.46276216e-01
 -1.77059509e-01 -1.74672916e-01 -2.77556895e-01 -1.95827626e-02
 -1.89333132e-01 -3.18000219e-02 -1.27345477e-01 -1.80966479e-01
 -9.17495017e-02 -2.48535039e-01 -4.67160043e-01 -9.18794772e-01
 -9.42576468e-01 -6.99885072e-01 -1.52542116e-02 -7.43870645e-01
 -4.21058702e-01 -1.46707502e+00 -9.83192021e-01 -1.13622068e-01
 -7.41046238e-01 -2.16806699e-01 -6.76887879e-02 -5.66394811e-01
 -1.00568625e-01 -6.73859938e-01 -8.17722928e-01 -8.04715609e-01
 -4.22844310e-01 -1.11368125e-02 -1.84059824e-01 -4.44603533e-01
 -2.85485780e-01 -1.46380637e+01 -3.67165137e-01 -3.93360720e+00
 -4.56738000e-01 -1.15260746e-01 -1.44744352e+00 -2.20557655e-02
 -4.86860012e-03 -1.67047004e-01 -3.96676236e-02 -1.13584669e-02
 -2.11621375e-02 -1.88222334e+00 -8.30893863e-02 -7.36612184e-01
 -5.24444292e-01 -1.67968937e-01 -4.44696252e-02 -5.67860981e-03
 -2.44382474e-01 -3.71154701e-01 -1.62363239e-01 -1.52373463e-02
 -3.92736624e-02 -2.65723329e-02 -5.67192119e-02 -3.65070912e-01
 -4.45249976e-02 -6.04055437e-02 -3.10376757e-02 -1.34880749e-02
 -2.55125343e-01 -3.89667026e-01 -2.11578257e-01 -2.42232027e-02
 -4.01709326e-01 -9.75338459e-02 -3.31269690e-01 -1.13211751e-01
 -5.82112419e-01 -1.88441263e-02 -1.38260104e-01 -1.71087784e-01
 -3.05998161e-02]
Pointwise PSIS: [ 1.82046693e-01  8.37305319e-02  2.08112910e-01  1.24415880e-01
  9.13515050e-03 -4.16602138e-02 -2.82087632e-01  2.58558743e-01
 -1.39527892e-01  2.34860363e-01  4.84934301e-02 -2.79645975e-02
  1.75937524e-01 -7.52425478e-02 -4.28808712e-01 -1.03425662e+00
 -1.03759327e+00 -6.82664645e-01  2.39458826e-01 -7.27820491e-01
 -9.77284701e-01 -2.41451617e+00 -1.91614493e+00 -3.36886741e-01
 -1.60415427e+00  1.01550578e-01  2.48987569e-01 -3.68926100e-01
  2.32694609e-01 -5.15240975e-01 -7.06257755e-01 -6.90512790e-01
 -1.71360741e-01  1.16669389e-01  1.45839477e-01 -2.15329785e-01
 -2.13030753e-03 -8.64318493e+00 -1.28740535e-01 -4.27572200e+00
 -2.68614316e-01 -3.03876689e-01 -2.40723182e+00  5.43449160e-02
  2.19753689e-01  1.30961591e-01 -7.74260234e-03  2.62379380e-01
  8.80869865e-02 -2.74335466e+00 -1.26935995e-01 -1.44242553e+00
 -1.09534573e+00 -3.38508583e-01  4.45232428e-02  2.64454057e-01
 -4.40400930e-01 -6.94217230e-01 -2.30852465e-01  2.73018265e-01
  1.20503204e-01  1.64991742e-01  7.62704232e-02 -3.36244745e-01
  1.16488585e-01  2.09019213e-01  2.53601679e-01  2.37404684e-01
 -1.59455211e-01 -4.01184172e-01 -8.50761713e-02  2.58496195e-01
```

```
-4.40636984e-01  1.23214129e-01 -3.50281505e-01  7.18286059e-02
-7.75232978e-01  2.58490351e-01  9.19337770e-03 -7.83435936e-02
 2.38731981e-01]
```



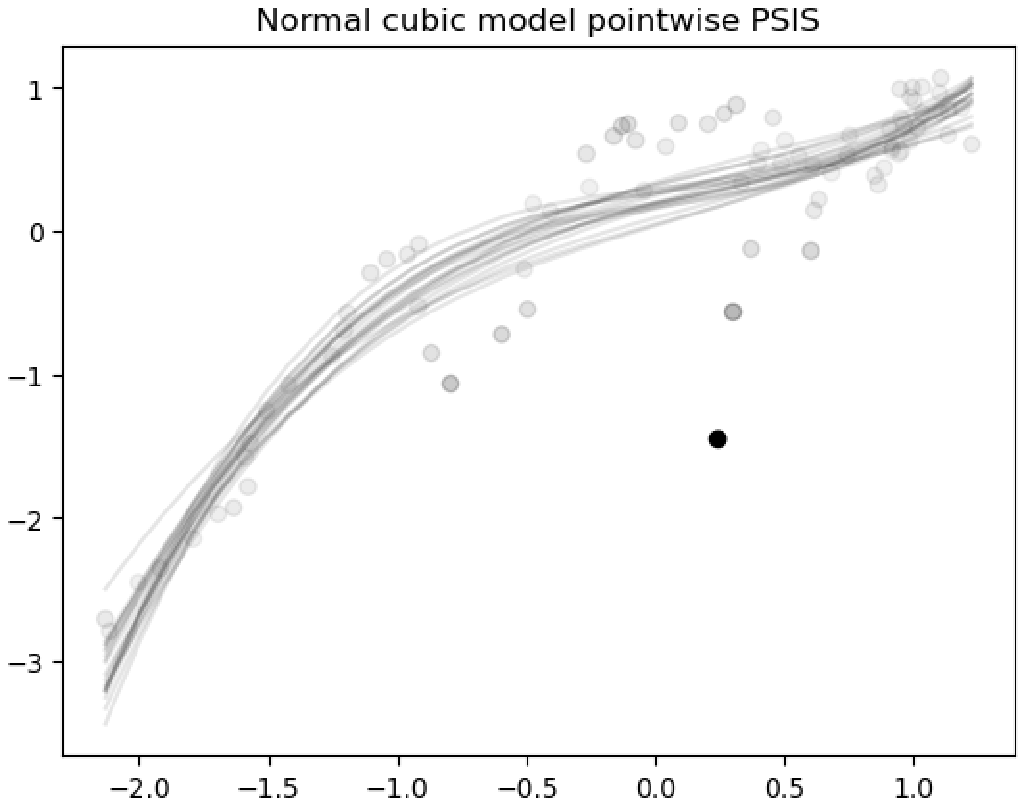


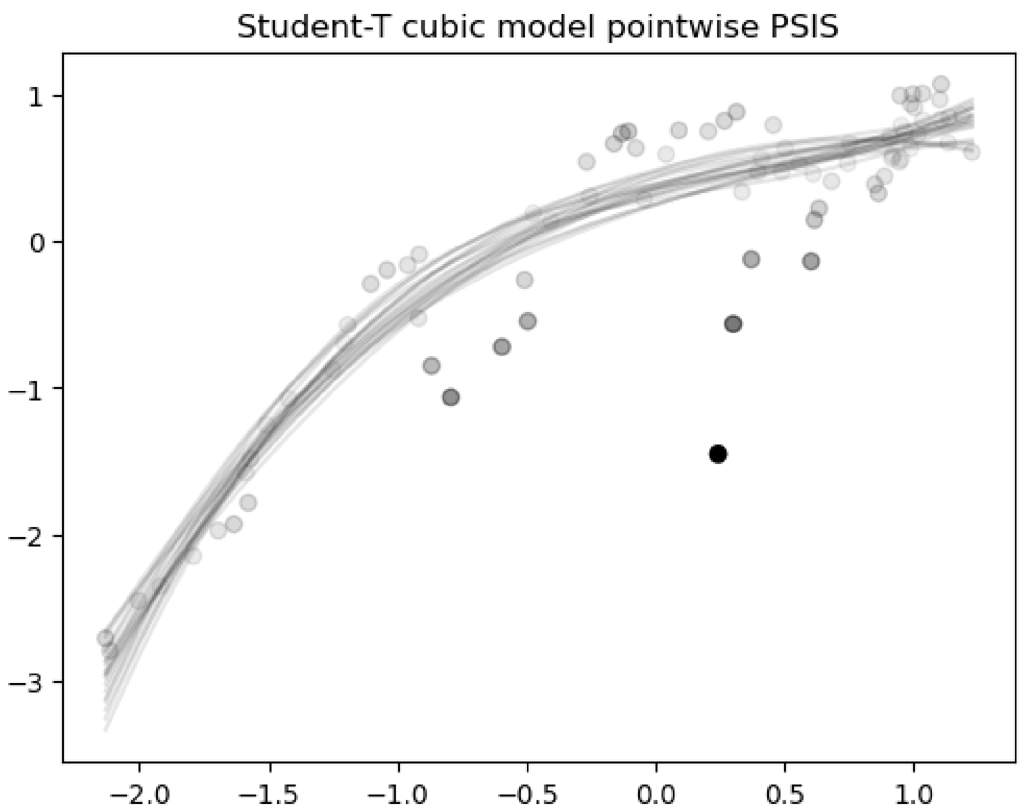
[175]:
```
plt.figure()
plt.title('Normal cubic model pointwise PSIS')
weights = az.loo(inference_3, pointwise=True).loo_i.values
print('Pointwise PSIS:', weights)
alpha = 1 - 0.95 * (weights - weights.min()) / (weights.max() - weights.min())
plt.scatter(data_x, data_y, color='black', alpha=alpha)
plt.plot(data_x, inference_3.posterior.mu[0, :20, :].transpose(), 'k-', alpha=0.
↪1)

plt.figure()
plt.title('Student-T cubic model pointwise PSIS')
weights = az.loo(inference_3t, pointwise=True).loo_i.values
print('Pointwise PSIS:', weights)
alpha = 1 - 0.95 * (weights - weights.min()) / (weights.max() - weights.min())
plt.scatter(data_x, data_y, color='black', alpha=alpha)
plt.plot(data_x, inference_3t.posterior.mu[0, :20, :].transpose(), 'k-', alpha=0.
↪1)
plt.show()
```

```
/Users/thananhthu/anaconda3/lib/python3.11/site-
packages/arviz/stats/stats.py:792: UserWarning: Estimated shape parameter of
```

```
Pareto distribution is greater than 0.70 for one or more samples. You should
consider using a more robust model, this is because importance sampling is less
likely to work well if the marginal posterior and LOO posterior are very
different. This is more likely to happen with a non-robust model and highly
influential observations.
  warnings.warn(

Pointwise PSIS: [-4.38870060e-01 -1.68005926e-01 -9.18836119e-02 -1.69988987e-02
 -1.03668219e-01 -1.95376084e-01 -3.99566196e-01 -6.86379403e-03
 -3.37834054e-01  1.78098610e-02  3.55109208e-03 -6.72996639e-03
  1.89870216e-02 -1.71686874e-02 -1.30773632e-01 -4.40330095e-01
 -4.66315959e-01 -3.04490197e-01 -2.71843888e-02 -3.49112013e-01
 -8.89321666e-01 -2.31219989e+00 -1.39595234e+00 -1.68202205e-01
 -9.67800244e-01 -1.15120143e-01 -6.51792959e-03 -6.08039725e-01
 -8.41031033e-02 -8.31490960e-01 -1.03707877e+00 -1.04718683e+00
 -5.96196322e-01  2.12374041e-02 -3.40360227e-01 -7.48547140e-01
 -5.75325036e-01 -1.40543518e+01 -7.24110142e-01 -3.31721009e+00
 -8.72718609e-01  1.93559680e-02 -9.93378705e-01  1.97198582e-02
 -4.15506942e-02 -4.14836888e-01  3.06060796e-02 -7.59503381e-02
  2.60600518e-02 -1.45763412e+00  2.63670713e-02 -4.41493988e-01
 -2.74762239e-01 -2.55980375e-02  3.16689643e-02  3.40172918e-03
 -1.60092919e-01 -2.95703660e-01 -1.07182105e-01  2.39975411e-02
  2.38565658e-03  1.49993400e-02 -2.97833143e-02 -3.20266811e-01
 -1.91227339e-02 -8.20281955e-03  2.14716292e-02  1.38012257e-02
 -1.64274672e-01 -2.81757674e-01 -1.10292716e-01  2.95888863e-02
 -2.42947414e-01 -1.97460912e-03 -1.10871607e-01  1.07431509e-02
 -2.87788112e-01 -4.93603038e-02  7.80465361e-03  2.15132624e-03
 -2.79735199e-01]
Pointwise PSIS: [-6.00623194e-01 -9.14445848e-02  4.58488880e-02  2.56309736e-01
  4.35994029e-02 -1.82976104e-01 -5.97944573e-01  2.31404829e-01
 -5.06551009e-01  3.00835480e-01  2.83792830e-01  2.70204063e-01
  2.82817688e-01  2.72522337e-01  7.47186822e-02 -4.34313597e-01
 -4.49361325e-01 -1.55831064e-01 -1.71321075e-02 -2.12757294e-01
 -1.70721539e+00 -3.09868290e+00 -2.41022903e+00 -6.18441285e-01
 -1.98054512e+00  2.40448680e-01  3.08261263e-01 -3.95995142e-01
  2.87252567e-01 -6.73133419e-01 -9.22087670e-01 -9.31164084e-01
 -3.53703406e-01  2.49093312e-01 -2.38607804e-04 -5.53798378e-01
 -3.34514114e-01 -8.23122130e+00 -5.46423418e-01 -3.95550191e+00
 -7.50723679e-01  3.07448995e-02 -2.05261148e+00  2.82436977e-01
  3.12261652e-01 -1.74524220e-01  2.50101025e-01  2.80023548e-01
  2.96533778e-01 -2.44155560e+00  1.51276689e-01 -1.11108355e+00
 -7.67706490e-01 -5.91040382e-02  2.32767051e-01  3.28113983e-01
 -3.17471913e-01 -6.04140226e-01 -1.48882980e-01  3.33663275e-01
  1.90037587e-01  2.35063019e-01  1.11989113e-01 -3.54141991e-01
  1.49122331e-01  2.68276858e-01  3.22869828e-01  2.70239648e-01
 -8.84053739e-02 -3.31585724e-01  1.46193498e-02  3.32371181e-01
 -2.95016894e-01  2.51147038e-01 -6.99209640e-02  2.69343374e-01
 -4.65906208e-01  2.17418774e-01  2.56890359e-01  2.47258815e-01
```

```
-8.64889938e-02]
```





6.4 Outlier Detection

[176]:
```
import pytensor.tensor as pt

with pm.Model() as outlier_model:

    # Observed variables
    x = pm.Data('x', data_x)
    y = pm.Data('y', data_y)

    # Linear regression
    c0 = pm.Uniform('c0', lower=0, upper=10)
    c1 = pm.Normal('c1', mu=0, sigma=1)
    c2 = pm.Normal('c2', mu=0, sigma=1)
    c3 = pm.Normal('c3', mu=0, sigma=1)
    mu = pm.Deterministic('mu', c0 + c1 * x + c2*x**2 + c3*x**3)

    # Noise parameters for inliers and outliers
    sigma = pm.Uniform('sigma', lower=0, upper=10)
    sigma_out = pm.HalfNormal('sigma_out', sigma=30)
    sigmas = pt.as_tensor_variable([sigma, sigma + sigma_out])
```

```python
    # In/out class assignment probability and indicators
    p = pm.Uniform('p', lower=0, upper=0.2)
    is_outlier = pm.Bernoulli('is_outlier', p=p, size=x.shape[0])

    pm.Normal('likelihood', mu=mu, sigma=sigmas[is_outlier], observed=y)
    outlier_inference = pm.sample(10000, tune=1000)
    pm.compute_log_likelihood(outlier_inference)

from IPython.display import Image
Image(pm.model_to_graphviz(outlier_model).render(format='png'))
```

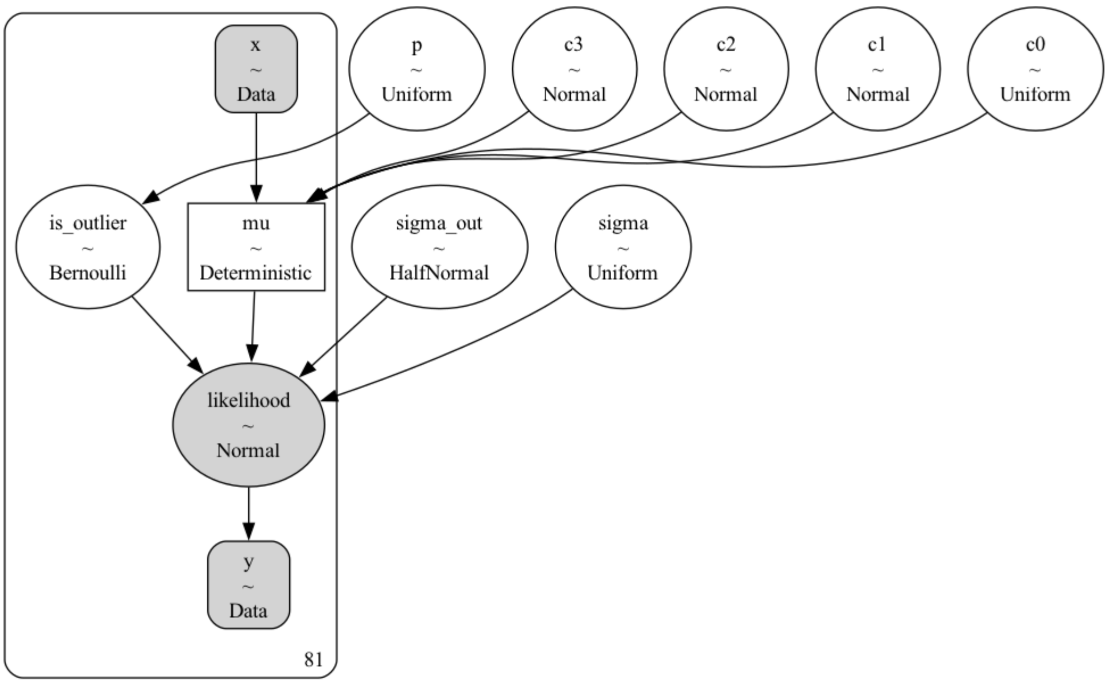
```
Multiprocess sampling (4 chains in 4 jobs)
CompoundStep
>NUTS: [c0, c1, c2, c3, sigma, sigma_out, p]
>BinaryGibbsMetropolis: [is_outlier]
```

```
Output()
```

```
Sampling 4 chains for 1_000 tune and 10_000 draw iterations (4_000 + 40_000
draws total) took 34 seconds.
There were 3 divergences after tuning. Increase `target_accept` or
reparameterize.
```

```
Output()
```

[176]:



6.4.1 Diagnostics

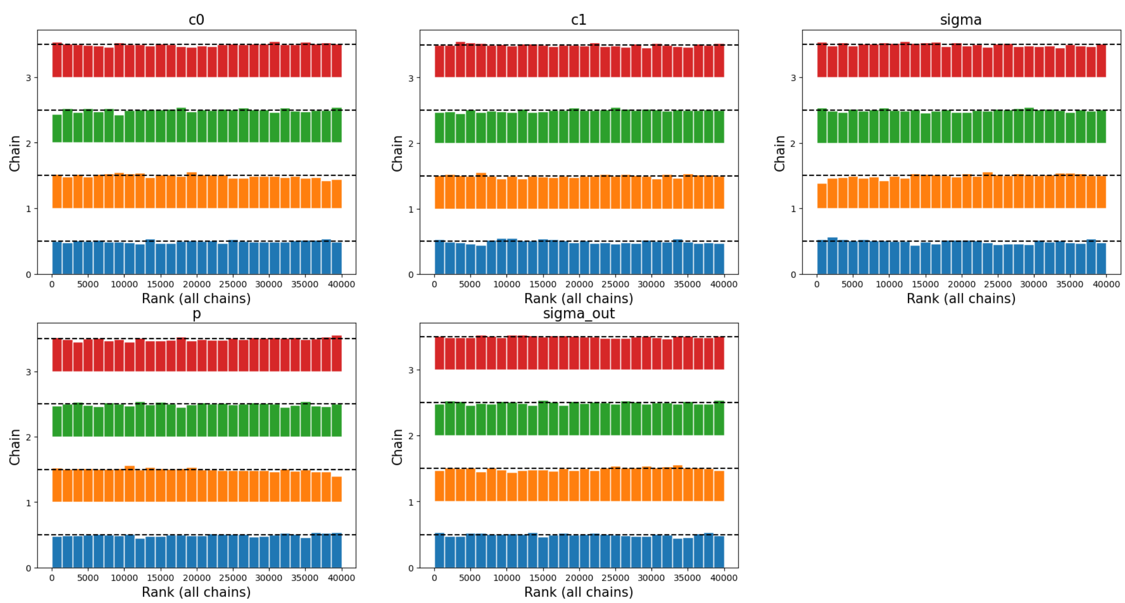
[178]:
```
az.summary(outlier_inference, var_names="~mu")
```

[178]:
```
                  mean     sd  hdi_5.5%  hdi_94.5%  mcse_mean  mcse_sd  \
c0               0.370  0.072     0.255      0.486      0.001    0.001
c1               0.468  0.080     0.343      0.599      0.001    0.001
c2              -0.248  0.070    -0.361     -0.137      0.001    0.001
c3               0.125  0.047     0.047      0.198      0.000    0.000
is_outlier[0]    0.049  0.216     0.000      0.000      0.002    0.001
...                ...    ...       ...        ...        ...      ...
is_outlier[79]   0.024  0.152     0.000      0.000      0.001    0.001
is_outlier[80]   0.030  0.172     0.000      0.000      0.001    0.001
p                0.080  0.053     0.003      0.161      0.001    0.001
sigma            0.278  0.040     0.213      0.340      0.001    0.000
sigma_out        3.364  6.962     0.211      7.327      0.084    0.060

                ess_bulk  ess_tail  r_hat
c0                4732.0   12122.0    1.0
c1               11063.0   18844.0    1.0
c2                8450.0   14656.0    1.0
c3               11702.0   16551.0    1.0
is_outlier[0]    17771.0   17771.0    1.0
...                  ...       ...    ...
```

```
is_outlier[79]   22122.0   22122.0    1.0
is_outlier[80]   23727.0   23727.0    1.0
p                 2594.0    6734.0    1.0
sigma             3339.0    7156.0    1.0
sigma_out         3528.0   11093.0    1.0

[88 rows x 9 columns]
```

[179]:
```
az.plot_rank(outlier_inference, var_names=['c0', 'c1', 'sigma', 'p',␣
↪'sigma_out'])
```

[179]:
```
array([[<Axes: title={'center': 'c0'}, xlabel='Rank (all chains)',
ylabel='Chain'>,
        <Axes: title={'center': 'c1'}, xlabel='Rank (all chains)',
ylabel='Chain'>,
        <Axes: title={'center': 'sigma'}, xlabel='Rank (all chains)',
ylabel='Chain'>],
       [<Axes: title={'center': 'p'}, xlabel='Rank (all chains)',
ylabel='Chain'>,
        <Axes: title={'center': 'sigma_out'}, xlabel='Rank (all chains)',
ylabel='Chain'>,
        <Axes: >]], dtype=object)
```

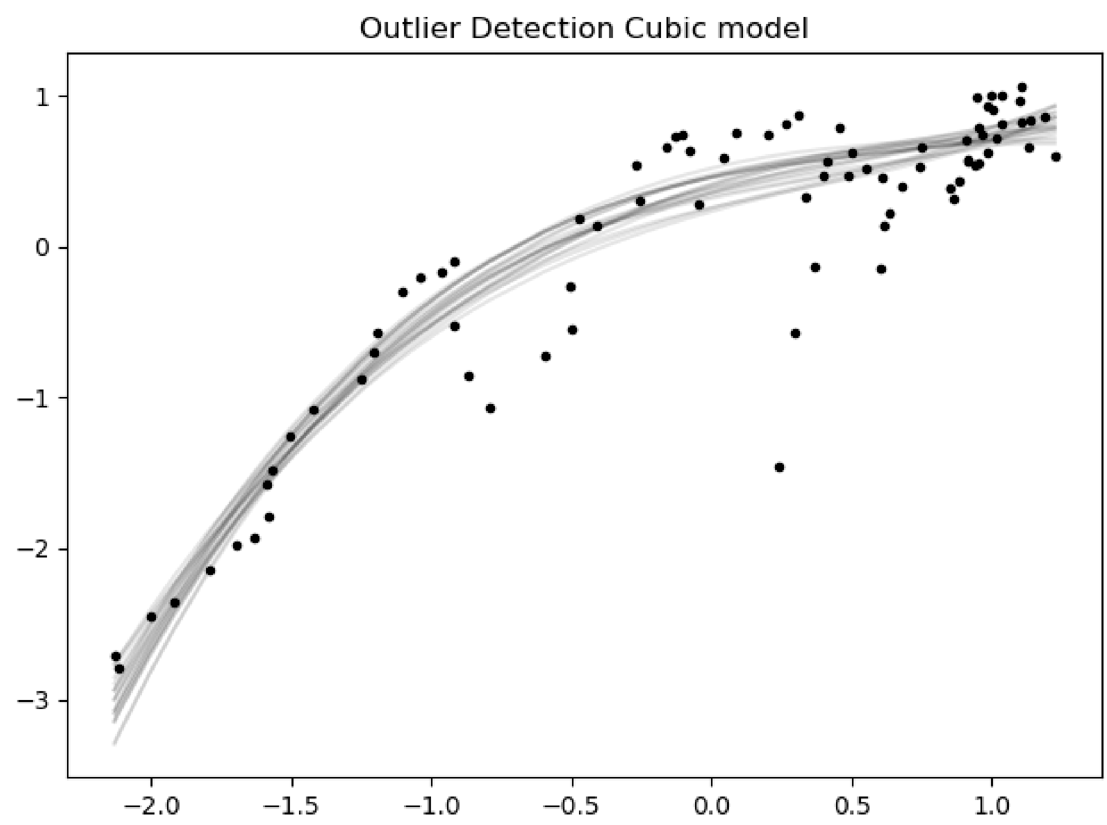


[180]:
```
# Sample from the posterior predictive distribution
with outlier_model:
    gdp_outlier = pm.sample_posterior_predictive(outlier_inference)
```

```
Sampling: [likelihood]
```

```
Output()
```

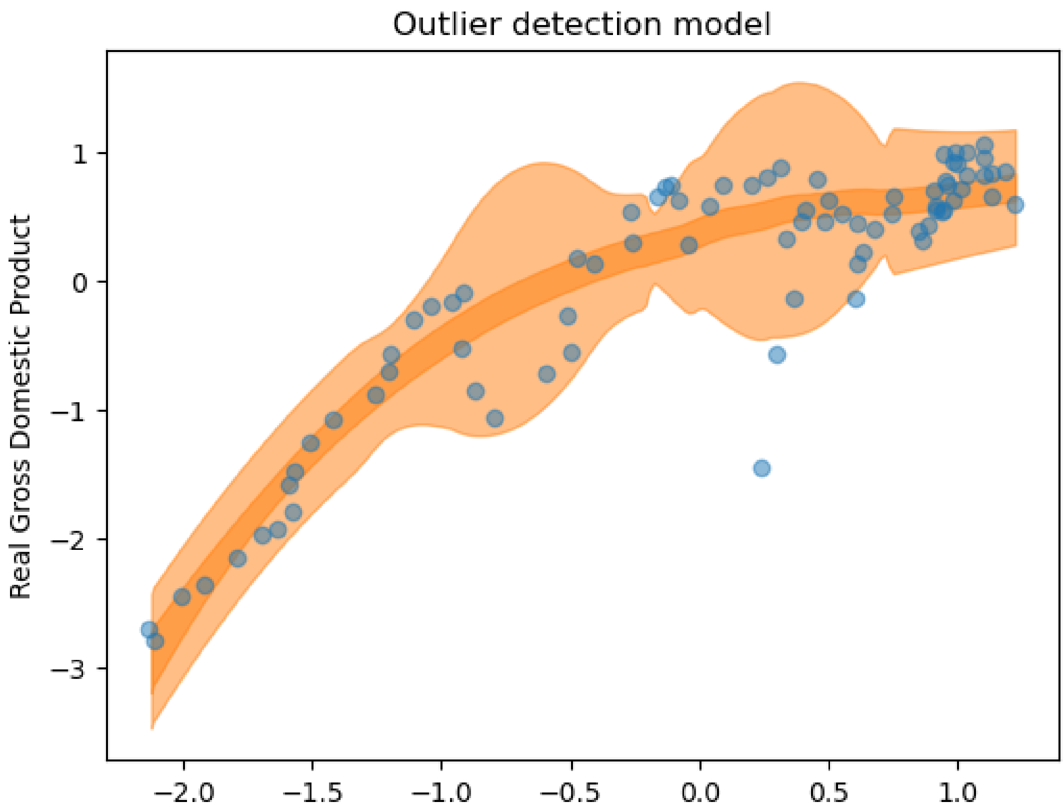
[181]:
```
plt.title('Outlier Detection Cubic model')
plt.plot(data_x, data_y, 'k.')
plt.plot(data_x, outlier_inference.posterior.mu[0, :20, :].transpose(), 'k-',␣
↪alpha=0.1)
plt.tight_layout()
plt.show()
```



[185]:
```
az.rcParams["stats.hdi_prob"] = 0.89

plt.title('Outlier detection model')
ax1 = az.plot_hdi(df['gov_exp_std'], outlier_inference.posterior["mu"])
az.plot_hdi(df['gov_exp_std'], gdp_outlier.posterior_predictive["likelihood"])
plt.scatter(df['gov_exp_std'], df['gdp_std'], alpha=0.5)
plt.ylabel("Real Gross Domestic Product")
```

```
/Users/thananhthu/anaconda3/lib/python3.11/site-packages/arviz/rcparams.py:345:
FutureWarning: stats.hdi_prob is deprecated since 0.18.0, use stats.ci_prob
instead
  warnings.warn(
```

[185]: Text(0, 0.5, 'Real Gross Domestic Product')



6.4.2 Model Comparison

[186]:
```python
print('Model comparison with PSIS')
df = az.compare({
    'Normal Cubic': inference_3,
    'Student-t Cubic': inference_3t,
    'Outlier Detection': outlier_inference},
    ic='loo')
df
```

```
Model comparison with PSIS
```

```
/Users/thananhthu/anaconda3/lib/python3.11/site-
packages/arviz/stats/stats.py:792: UserWarning: Estimated shape parameter of
Pareto distribution is greater than 0.70 for one or more samples. You should
```

```
consider using a more robust model, this is because importance sampling is less
likely to work well if the marginal posterior and LOO posterior are very
different. This is more likely to happen with a non-robust model and highly
influential observations.
  warnings.warn(
/Users/thananhthu/anaconda3/lib/python3.11/site-
packages/arviz/stats/stats.py:1039: RuntimeWarning: overflow encountered in exp
  weights = 1 / np.exp(len_scale - len_scale[:, None]).sum(axis=1)
/Users/thananhthu/anaconda3/lib/python3.11/site-
packages/arviz/stats/stats.py:792: UserWarning: Estimated shape parameter of
Pareto distribution is greater than 0.70 for one or more samples. You should
consider using a more robust model, this is because importance sampling is less
likely to work well if the marginal posterior and LOO posterior are very
different. This is more likely to happen with a non-robust model and highly
influential observations.
  warnings.warn(
```

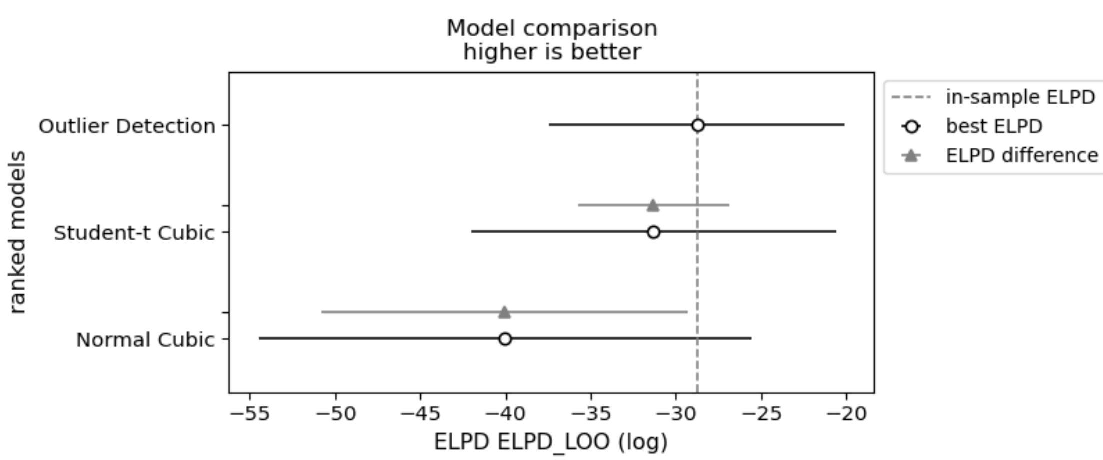
[186]:

	rank	elpd_loo	p_loo	elpd_diff	weight	se
Outlier Detection	0	-28.747326	15.571872	0.000000	0.733101	8.665897
Student-t Cubic	1	-31.299420	7.325974	2.552095	0.000000	10.702758
Normal Cubic	2	-40.041858	7.268029	11.294533	0.266899	14.458054

	dse	warning	scale
Outlier Detection	0.000000	True	log
Student-t Cubic	4.438696	False	log
Normal Cubic	10.761657	True	log

[187]:

```python
az.plot_compare(df)
plt.title('Model comparison\nhigher is better')
plt.xlabel('ELPD ' + df.columns[1].upper() + ' (log)')
plt.legend(['in-sample ELPD', 'best ELPD', 'ELPD difference', 'ELPD'],␣
 ↪loc='upper left', bbox_to_anchor=(1.0, 1.0))
plt.show()
```



[189]:
```python
print('Model comparison with WAIC')
df = az.compare({
    'Normal Cubic': inference_3,
    'Student-t Cubic': inference_3t,
    'Outlier Detection': outlier_inference},
    ic='loo')
df
```

```
Model comparison with WAIC

/Users/thananhthu/anaconda3/lib/python3.11/site-
packages/arviz/stats/stats.py:792: UserWarning: Estimated shape parameter of
Pareto distribution is greater than 0.70 for one or more samples. You should
consider using a more robust model, this is because importance sampling is less
likely to work well if the marginal posterior and LOO posterior are very
different. This is more likely to happen with a non-robust model and highly
influential observations.
  warnings.warn(
/Users/thananhthu/anaconda3/lib/python3.11/site-
packages/arviz/stats/stats.py:1039: RuntimeWarning: overflow encountered in exp
  weights = 1 / np.exp(len_scale - len_scale[:, None]).sum(axis=1)
/Users/thananhthu/anaconda3/lib/python3.11/site-
packages/arviz/stats/stats.py:792: UserWarning: Estimated shape parameter of
Pareto distribution is greater than 0.70 for one or more samples. You should
consider using a more robust model, this is because importance sampling is less
likely to work well if the marginal posterior and LOO posterior are very
different. This is more likely to happen with a non-robust model and highly
influential observations.
  warnings.warn(
```

[189]:
```
                    rank   elpd_loo      p_loo  elpd_diff    weight         se  \
Outlier Detection      0 -28.747326  15.571872   0.000000  0.733101   8.665897
Student-t Cubic        1 -31.299420   7.325974   2.552095  0.000000  10.702758
Normal Cubic           2 -40.041858   7.268029  11.294533  0.266899  14.458054

                          dse  warning scale
Outlier Detection    0.000000     True   log
Student-t Cubic      4.438696    False   log
Normal Cubic        10.761657     True   log
```

[190]:
```python
az.plot_compare(df)
# Add these labels since they are not in the latest release (scheduled for the␣
 ↪next one)
plt.title('Model comparison\nhigher is better')
plt.xlabel('ELPD ' + df.columns[1].upper() + ' (log)')
```

```
plt.legend(['in-sample ELPD', 'best ELPD', 'ELPD difference', 'ELPD'],␣
↪loc='upper left', bbox_to_anchor=(1.0, 1.0))
plt.show()
```