GENERAL

python quantile regression

Published: September 3, 2026 | Index ID: #-

When building predictive models, most practitioners default to ordinary least squares (OLS) regression because it is simple, fast, and well understood. However, OLS only tells you about the mean of the response variable given the predictors. In many real world scenarios, you care about the entire distribution of the outcome—especially the tails. Quantile regression fills that gap by estimating conditional quantiles, such as the median or the 90th percentile, directly from the data. This article dives deep into Python quantile regression, covering theory, practical implementation, and real world applications.

What Is Quantile Regression?

Quantile regression, introduced by Koenker and Bassett in 1978, extends linear regression to estimate any conditional quantile of the response variable. While OLS minimizes the sum of squared residuals to estimate the mean, quantile regression minimizes a weighted absolute loss function that focuses on a specific quantile. For the 50th percentile (median), the loss is symmetric; for the 90th percentile, the loss penalizes under predictions more heavily than over predictions.

Mathematically, for a quantile level <B , ƒ Ã ’Â F†R ö&¦V7F—`e function is:

minimize :6’ Á<B‡–’ b †“²),

where < Ä(u) = u(<B b ·R

Key Differences From OLS

- Target Statistic: OLS estimates the mean; quantile regression estimates a chosen quantile.
- Loss Function: OLS uses squared loss; quantile regression uses asymmetric absolute loss.
- Robustness: Quantile regression is less sensitive to outliers, especially when estimating higher or lower quantiles.
- Interpretation: Coefficients reflect how predictors shift the chosen quantile, not just the mean.

Why Use Quantile Regression?

Quantile regression shines in situations where the relationship between predictors and the outcome changes across the distribution. Consider the following scenarios:

1. Income Inequality: Predicting the 90th percentile of income helps policymakers target the wealthiest households.
2. Insurance Premiums: Insurers often model the 95th percentile of claim costs to set reserves.
3. Environmental Studies: Estimating extreme temperature quantiles aids climate risk assessment.
4. Real Estate: Understanding how location affects the upper tail of house prices informs investment decisions.
5. Machine Learning: Quantile regression can serve as a robust baseline for predicting uncertainty intervals.

In each case, focusing solely on the mean would obscure critical tail behavior. Quantile regression provides a richer, more actionable picture.

Python Libraries for Quantile Regression

Python offers several libraries that implement quantile regression. The most popular ones are **statsmodels** and **scikit learn**Additionally, **QuantileTransformer** from scikit learn can be used for preprocessing, while**PyTorch** and **TensorFlow** enable deep learning approaches.

Statsmodels

Statsmodels provides a dedicated QuantReg class that follows the traditional statistical framework. It supports:

- Multiple quantile levels in a single fit.
- Standard errors and hypothesis tests.
- Robust covariance estimation.
- Convenient summary tables.

Scikit Learn

Scikit learn's QuantileRegressor is built on top of the SGDRegressor with a quantile loss. It offers:

- Incremental learning for large datasets.
- Regularization (L1/L2) to prevent overfitting.
- Easy integration into scikit learn pipelines.

Other Options

- PyTorch & TensorFlow: Custom loss functions allow neural networks to perform quantile regression.
- QuantileTransformer: Transforms features to a uniform distribution before regression.
- QuantileRegressor from the statsmodels GLM module: For generalized linear models with quantile loss.

Preparing Your Data for Quantile Regression

Before fitting a model, clean and preprocess the data carefully. The steps mirror those for any regression task but with a few nuances:

1. Handle Missing Values: Drop or impute missing entries. Imputation should respect the quantile structure—median imputation works well for skewed data.
2. Feature Scaling: Standardization is not mandatory, but it helps with regularization. For tree based models, scaling is unnecessary.
3. Encode Categorical Variables: Use one hot encoding or ordinal encoding if the categories have an inherent order.
4. Outlier Detection: Unlike OLS, quantile regression is robust, but extreme outliers can still distort higher quantiles. Consider trimming or winsorizing if necessary.
5. Train Test Split: Keep the split random and stratified if the target distribution is heavily skewed.

Step by Step: Implementing Quantile Regression in Python

Below is a complete example that walks through data loading, preprocessing, model fitting, and interpretation. We use the statsmodels library for its rich statistical output.

1. Import Libraries

```
import numpy as np

import pandas as pd

import statsmodels.api as sm

import matplotlib.pyplot as plt
```

```python
import seaborn as sns
```

2. Load Sample Data

We’ll use the well known Boston Housing dataset, but replace the target variable with a synthetic skewed distribution to illustrate tail estimation.

```python
from sklearn.datasets import load_boston

data = load_boston()

df = pd.DataFrame(data.data, columns=data.feature_names)

df['MEDV'] = data.target

# Introduce skewness

df['MEDV'] = df['MEDV'] ** 1.5
```

3. Preprocess Features

```python
X = df.drop('MEDV', axis=1)

y = df['MEDV']

# Add constant for intercept

X_const = sm.add_constant(X)
```

4. Fit Quantile Models

We’ll estimate the 25th, 50th (median), and 75th percentiles.

```python
taus = [0.25, 0.5, 0.75]

models = {tau: sm.QuantReg(y, X_const).fit(q=tau) for tau in taus}
```

5. Summarize Results

Print the summary for each quantile.

```python
for tau, model in models.items():

    print(f'\nQuantile: {tau}')

    print(model.summary())
```

6. Visualize Predictions

Plot the fitted quantile lines against the actual data.

```python
plt.figure(figsize=(10,6))

sns.scatterplot(x='RM', y='MEDV', data=df, label='Data', alpha=0.5)

for tau, model in models.items():

    pred = model.predict(X_const)


    plt.plot(df['RM'], pred, label=f'Quantile {tau}')

plt.xlabel('Average number of rooms per dwelling (RM)')
```

```
plt.ylabel('Median value of owner  occupied homes (MEDV)')

plt.title('Quantile Regression Fit on Boston Housing')

plt.legend()

plt.show()
```

Interpreting Quantile Regression Coefficients

Unlike OLS, where coefficients represent the change in the mean response per unit change in a predictor, quantile coefficients describe the shift in a specific quantile. For example, a coefficient of 0.3 for the predictor RM at the 90th percentile means that each additional room increases the 90th percentile of home value by $0.3k.

When comparing coefficients across quantiles, pay attention to:

- Magnitude Differences: A larger coefficient at the upper quantile indicates a stronger impact on high value homes.
- Sign Changes: A predictor may have a positive effect on the median but a negative effect on the lower tail.
- Statistical Significance: Use the t statistics and p values provided by statsmodels. A predictor significant in the median but not in the tails may not influence extreme outcomes.

Confidence Intervals

Statsmodels returns robust standard errors, enabling the construction of confidence intervals for each quantile:

```
conf_int = model.conf_int(alpha=0.05)
```

These intervals can be plotted alongside the coefficient estimates to assess uncertainty.

Diagnostics and Model Checking

Just like any regression model, quantile regression benefits from diagnostic checks.

Residual Analysis

Plot residuals versus fitted values for each quantile. Ideally, residuals should be randomly scattered without patterns.

Influence Measures

While statsmodels does not provide Cook’s distance for quantile regression by default, you can compute leverage and influence using

Tags: #python #quantile #regression