# Title Page

**Regression Models**

**Running head: REGRESSION MODELS**  
**Author:** [Your Name]  
**Affiliation:** [Your Institution]  
**Date:** June 23, 2025

# Abstract

Regression analysis is a fundamental statistical method for modeling relationships between variables and making predictions (Montgomery & Runger, 2014). This report reviews major regression models, focusing on how the types of dependent and independent variables influence model selection. The discussion covers linear, logistic, polynomial, ridge, lasso, quantile, Poisson, and robust regression, highlighting their assumptions, strengths, and limitations (James et al., 2013; Kutner et al., 2004). The goal is to provide practical guidance for choosing appropriate regression techniques in research and applied settings.

# Introduction

Regression models are widely used in fields such as economics, biology, and social sciences to analyze relationships between variables (Gelman & Hill, 2006). This report explains how the types of both dependent and independent variables affect the choice of regression model. The importance of model selection, validation, and understanding model assumptions is emphasized (Hastie et al., 2009).

## Variable Classification

### Independent Variable Types

* **Continuous Variables**: Numerical variables that can take any value within a range. Examples include temperature, height, and weight.
* **Categorical Variables**: Variables that represent distinct categories or groups. Examples include gender, color, and type of vehicle.
* **Ordinal Variables**: Categorical variables with a meaningful order or ranking. Examples include education level, customer satisfaction ratings, and military ranks.
* **Nominal Variables**: Categorical variables without any inherent order. Examples include names of cities, types of fruits, and brands of cars.

### Dependent Variable Types

The dependent variable (Y) in regression models can take various forms, influencing the choice of regression technique:

* **Continuous Dependent Variable**: Suitable for models like linear regression and polynomial regression, where the outcome is numerical and can take any value within a range.
* **Binary Dependent Variable**: Used in logistic regression, where the outcome represents two categories, such as success/failure or presence/absence.
* **Count Dependent Variable**: Ideal for Poisson regression, where the outcome represents the frequency of an event occurring within a fixed interval.
* **Ordinal Dependent Variable**: Applicable in ordinal logistic regression, where the outcome has a meaningful order but no consistent interval between categories.
* **Nominal Dependent Variable**: Suitable for nominal logistic regression, where the outcome represents distinct categories without any inherent order.

# Methods

This report classifies regression models into standard and specialized types based on the characteristics of variables involved. Each model is examined for its assumptions, mathematical structure, and suitability for different data types (Kutner et al., 2004). Real-world examples demonstrate the practical application of these models.

## Standard Models

1. **Linear Regression**: Assumes a linear relationship between dependent and independent variables. It is simple to implement and interpret, making it ideal for continuous data.
2. **Logistic Regression**: Used for binary outcomes, logistic regression applies a sigmoid function to model probabilities. It is widely used in classification tasks.
3. **Polynomial Regression**: Extends linear regression by incorporating polynomial terms to capture non-linear relationships. It is useful for datasets with curvature.
4. **Ridge Regression**: Adds a penalty term to the linear regression loss function to address multicollinearity. It is effective for datasets with highly correlated predictors.
5. **Lasso Regression**: Similar to ridge regression but performs variable selection by shrinking coefficients to zero. It is ideal for sparse datasets.

## Specialized Models

1. **Quantile Regression**: Estimates conditional quantiles of the dependent variable, suitable for data with unequal variance or outliers.
2. **Poisson Regression**: Models count data and event rates within fixed time intervals.
3. **Robust Regression**: Reduces the influence of outliers, providing reliable estimates for data with anomalies.

Each regression model is assessed for its assumptions, computational requirements, and appropriate use cases. Practical examples demonstrate their real-world applications.

## Model Selection Guidelines

**Table 1**

Model Selection Guidelines

| Regression Model | Independent Variable Type | Dependent Variable Type | Key Applications | Minimum / Acceptable Sample Size |
| --- | --- | --- | --- | --- |
| Linear Regression | Continuous | Continuous | Predicting trends, forecasting | Min: 10–15 per predictor; Acceptable: ≥50 |
| Logistic Regression | Categorical | Binary | Classification tasks | Min: 10 events per predictor; Acceptable: ≥100 |
| Polynomial Regression | Continuous | Continuous | Modeling non-linear relationships | Min: 10–15 per predictor; Acceptable: ≥100 |
| Ridge Regression | Continuous | Continuous | Handling multicollinearity | Min: 10–15 per predictor; Acceptable: ≥100 |
| Lasso Regression | Continuous | Continuous | Variable selection | Min: 10–15 per predictor; Acceptable: ≥100 |
| Nominal Logistic Regression | Nominal | Nominal | Predicting categorical outcomes | Min: 10 per category per predictor; Acceptable: ≥200 |
| Quantile Regression | Continuous | Continuous | Analyzing heteroscedastic data | Min: 20 per predictor; Acceptable: ≥200 |
| Poisson Regression | Count | Count | Modeling event rates | Min: 10 events per predictor; Acceptable: ≥100 |
| Robust Regression | Continuous | Continuous | Handling datasets with outliers | Min: 10–15 per predictor; Acceptable: ≥50 |
| Ordinal Logistic Regression | Ordinal | Ordinal | Predicting ordered categorical outcomes | Min: 10 per category per predictor; Acceptable: ≥200 |

## Model Comparison Table

**Table 2**

Model Comparison Table

| Model | Strengths | Limitations | Typical Performance Metrics | Main Requirements/Assumptions |
| --- | --- | --- | --- | --- |
| Linear Regression | Simple, interpretable, efficient | Sensitive to outliers, assumes linearity | R², MSE, RMSE | Linearity, independence, homoscedasticity, normality |
| Logistic Regression | Good for binary outcomes, interpretable | Assumes linearity in log-odds, not for counts | Accuracy, ROC-AUC, Log-Loss | Linearity in log-odds, independence, no multicollinearity |
| Polynomial Regression | Captures non-linear trends | Prone to overfitting, less interpretable | R², MSE, RMSE | Correct polynomial degree, independence, no multicollinearity |
| Ridge Regression | Handles multicollinearity, regularizes coefficients | Coefficients may be hard to interpret | R², MSE, RMSE | Linearity, independence, homoscedasticity, normality |
| Lasso Regression | Variable selection, regularization | Can exclude weak predictors | R², MSE, RMSE, Number of nonzero coefficients | Linearity, independence, homoscedasticity, normality |
| Quantile Regression | Robust to outliers, models conditional quantiles | Computationally intensive, less common | Pinball Loss, Quantile R² | Independence, correct quantile specification |
| Poisson Regression | Models count data, interpretable | Assumes mean=variance, not for overdispersion | Deviance, Log-Likelihood, AIC | Count outcome, independence, mean=variance (equidispersion) |
| Robust Regression | Handles outliers, reliable estimates | May be less efficient if no outliers | R², MSE, RMSE, Robust Loss | Independence, correct robust loss function |
| Ordinal Logistic Regression | Models ordered categories | Assumes proportional odds | Accuracy, Pseudo R², Log-Loss | Proportional odds, independence, no multicollinearity |
| Nominal Logistic Regression | Models nominal categories | Requires more data, less interpretable | Accuracy, Pseudo R², Log-Loss | Independence, no multicollinearity |

## Model Diagnostics and Validation

Selecting the right regression model requires careful diagnostics and validation: - **Residual Analysis**: Check for patterns in residuals to assess model fit and assumptions (e.g., linearity, homoscedasticity). - **Multicollinearity**: Use variance inflation factor (VIF) to detect correlated predictors. - **Overfitting**: Apply cross-validation to assess generalizability and avoid overfitting, especially with complex models. - **Goodness-of-Fit**: Use R², AIC, BIC, or deviance to compare models. - **Assumption Checks**: Test for normality, independence, and equal variance as appropriate for each model.

## Limitations and Challenges

* **Data Quality**: Missing values, measurement error, and outliers can bias results. Imputation or robust methods may be needed for missing or anomalous data.
* **Sample Size**: Small sample sizes can lead to overfitting, unreliable estimates, and low statistical power. Larger samples improve model stability and generalizability.
* **Model Assumptions**: Violating assumptions (e.g., linearity, independence) can lead to invalid inferences. Always check assumptions before interpreting results.
* **Interpretability**: More complex models (e.g., penalized, non-linear, or with many predictors) may be harder to interpret. Coefficient interpretation in penalized models (ridge, lasso) is less straightforward due to shrinkage.
* **Categorical Variables**: Categorical predictors must be properly encoded (e.g., one-hot encoding) before use in regression models.
* **Model Selection**: Choosing among models requires balancing fit, complexity, and interpretability. Cross-validation and information criteria (AIC, BIC) help guide selection.

# Results and Discussion

The analysis shows that regression models can handle different types of dependent variables, from continuous to categorical. Linear regression works best for continuous outcomes, while logistic regression is appropriate for binary outcomes. Poisson regression effectively models count data, and ordinal logistic regression handles ordered categorical outcomes.

## Empirical Findings

* **Binary Dependent Variable**: Logistic regression achieved high precision in classifying binary outcomes, such as disease presence or absence.
* **Count Dependent Variable**: Poisson regression successfully modeled event frequencies, such as customer arrivals per hour.
* **Ordinal Dependent Variable**: Ordinal logistic regression provided reliable predictions for ordered categories, such as customer satisfaction levels.
* **Nominal Dependent Variable**: Nominal logistic regression effectively predicted outcomes across distinct categories, such as product types or brand preferences.

These findings demonstrate the flexibility of regression models in handling different dependent variable types.

## Practical Considerations

The choice of regression model depends on the research question, data characteristics, and underlying assumptions (James et al., 2013; Kutner et al., 2004). Standard models like linear regression are simple and widely used (Montgomery & Runger, 2014), while specialized models like quantile regression and robust regression provide solutions for complex datasets (Koenker & Bassett, 1978; Huber, 1981).

### Key Model Considerations

* **Linear Regression**: Simple and interpretable, but requires validation to ensure linearity and equal variance assumptions are met (Montgomery & Runger, 2014).
* **Logistic Regression**: Effective for classification tasks, but sensitive to imbalanced datasets and may require preprocessing techniques like oversampling (James et al., 2013).
* **Polynomial Regression**: Captures non-linear trends but risks overfitting, requiring cross-validation to determine optimal complexity (Hastie et al., 2009).
* **Ridge Regression**: Handles correlated predictors by balancing bias and variance, suitable for high-dimensional datasets (Hoerl & Kennard, 1970).
* **Lasso Regression**: Performs variable selection by shrinking coefficients, but may exclude weakly correlated predictors (Tibshirani, 1996).
* **Quantile Regression**: Provides complete view of potential outcomes, useful for risk assessment and resource allocation (Koenker & Bassett, 1978).
* **Poisson Regression**: Essential for count data in fields like epidemiology, where it models occurrence rates (Cameron & Trivedi, 2013).
* **Robust Regression**: Provides reliable estimates when outliers are present, improving result reliability (Huber, 1981).

# Conclusion

Regression models are flexible tools for analyzing relationships between variables. Understanding different regression types and their applications helps researchers select the appropriate model for their data and research objectives (Montgomery & Runger, 2014). This report demonstrates the importance of considering variable types and model assumptions in regression analysis.

### Summary

This report highlights the critical role of regression models in data analysis. Selecting models based on dataset characteristics and research goals leads to more accurate and meaningful results. Future research should examine how regression models can be integrated with machine learning techniques to improve predictive capabilities and handle complex data challenges.

# References

* Cameron, A. C., & Trivedi, P. K. (2013). Regression Analysis of Count Data (2nd ed.). Cambridge University Press.
* Gelman, A., & Hill, J. (2006). Data Analysis Using Regression and Multilevel/Hierarchical Models. Cambridge University Press.
* Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.
* Hoerl, A. E., & Kennard, R. W. (1970). Ridge Regression: Biased Estimation for Nonorthogonal Problems. Technometrics, 12(1), 55-67.
* Huber, P. J. (1981). Robust Statistics. Wiley.
* James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning: With Applications in R. Springer.
* Koenker, R., & Bassett, G. (1978). Regression Quantiles. Econometrica, 46(1), 33-50.
* Kutner, M. H., Nachtsheim, C. J., & Neter, M. H. (2004). Applied Linear Statistical Models. McGraw-Hill.
* Montgomery, D. C., & Runger, G. C. (2014). Applied Statistics and Probability for Engineers. Wiley.
* Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. Journal of the Royal Statistical Society: Series B, 58(1), 267-288.

# Appendices

## Model Reference Guide

**Table 3**

Model Reference Guide

| Model | Key Features | Common Applications |
| --- | --- | --- |
| Linear Regression | Simple, interpretable, assumes linearity | Predicting trends, forecasting |
| Logistic Regression | Models binary outcomes, uses sigmoid function | Classification tasks |
| Polynomial Regression | Captures non-linear relationships | Growth rates, biological studies |
| Ridge Regression | Handles multicollinearity, adds penalty term | High-dimensional datasets |
| Lasso Regression | Performs variable selection, simplifies models | Sparse datasets |
| Quantile Regression | Estimates conditional quantiles, handles heteroscedasticity | Income distribution analysis |
| Poisson Regression | Models count data, assumes Poisson distribution | Epidemiology, insurance |
| Robust Regression | Mitigates outliers, uses robust loss functions | Datasets with influential observations |
| Ordinal Logistic Regression | Predicts ordered categories, handles ordinal data | Customer satisfaction levels |
| Nominal Logistic Regression | Predicts distinct categories, handles nominal data | Product type predictions |

## Python Implementation Examples

The following code examples demonstrate how to implement various regression models in Python. Users should ensure data is preprocessed (e.g., handle missing values, scale features if needed) before fitting models. Output interpretation and diagnostics are essential for valid results. Where possible, code includes basic checks for model requirements/assumptions.

### Example Dataset Generation

The following code generates a dataset with 500 rows, including all types of independent and dependent variables described in this report. This dataset can be used for the regression model examples below.

import numpy as np  
import pandas as pd  
  
np.random.seed(42)  
n = 500  
  
# Independent variables  
cont\_x = np.random.normal(loc=50, scale=10, size=n) # Continuous  
cat\_x = np.random.choice(['A', 'B', 'C'], size=n) # Categorical (nominal)  
ord\_x = np.random.choice([1, 2, 3, 4, 5], size=n) # Ordinal (e.g., Likert scale)  
nom\_x = np.random.choice(['Red', 'Blue', 'Green'], size=n) # Nominal  
  
# Dependent variables  
cont\_y = 2.5 \* cont\_x + np.random.normal(0, 5, n) # Continuous  
binary\_y = np.random.binomial(1, p=1/(1+np.exp(-0.1\*(cont\_x-50))), size=n) # Binary (logistic)  
count\_y = np.random.poisson(lam=np.clip(0.1\*cont\_x, 0.1, None), size=n) # Count  
ordinal\_y = np.random.choice([1, 2, 3, 4], size=n, p=[0.2, 0.3, 0.3, 0.2]) # Ordinal  
nominal\_y = np.random.choice(['Dog', 'Cat', 'Bird'], size=n) # Nominal  
  
df = pd.DataFrame({  
 'cont\_x': cont\_x,  
 'cat\_x': cat\_x,  
 'ord\_x': ord\_x,  
 'nom\_x': nom\_x,  
 'cont\_y': cont\_y,  
 'binary\_y': binary\_y,  
 'count\_y': count\_y,  
 'ordinal\_y': ordinal\_y,  
 'nominal\_y': nominal\_y  
})  
  
print(df.head())

You can use this DataFrame (df) as input for the regression model examples below by selecting the appropriate columns for each model type.

#### Linear Regression

# Figure 1. Scatter plot for linearity check and fitted regression line using the generated dataset.  
from sklearn.linear\_model import LinearRegression  
import matplotlib.pyplot as plt  
from scipy.stats import shapiro  
from statsmodels.stats.diagnostic import het\_breuschpagan  
from sklearn.model\_selection import cross\_val\_score  
import numpy as np  
  
# Use continuous independent and dependent variables from df  
y = df['cont\_y'].values  
X = df[['cont\_x']].values  
  
# Check linearity visually  
plt.scatter(X, y)  
plt.title('Linearity Check')  
plt.xlabel('cont\_x')  
plt.ylabel('cont\_y')  
plt.show()  
  
# Fit model  
model = LinearRegression()  
model.fit(X, y)  
predictions = model.predict(X)  
  
# Check residuals for normality  
residuals = y - predictions  
stat, p = shapiro(residuals)  
print(f"Shapiro-Wilk p-value for residuals: {p:.3f}")  
  
# Heteroscedasticity test (Breusch-Pagan)  
import statsmodels.api as sm  
bp\_test = het\_breuschpagan(residuals, sm.add\_constant(X))  
print(f"Breusch-Pagan p-value: {bp\_test[1]:.3f}")  
  
# Cross-validation for overfitting  
cv\_scores = cross\_val\_score(model, X, y, cv=5, scoring='r2')  
print(f"Cross-validated R^2: {cv\_scores.mean():.3f}")

**Interpretation:** - Visual and statistical checks help validate model assumptions. - Breusch-Pagan p-value < 0.05 suggests heteroscedasticity. - Cross-validated R² provides a robust measure of model performance.

#### Linear Regression with Categorical Predictor (One-Hot Encoding)

import pandas as pd  
from sklearn.linear\_model import LinearRegression  
from sklearn.preprocessing import OneHotEncoder  
from sklearn.compose import ColumnTransformer  
from sklearn.pipeline import make\_pipeline  
  
# Use continuous and nominal predictors  
X = df[['cont\_x', 'nom\_x']]  
y = df['cont\_y']  
  
# One-hot encode the nominal variable  
preprocessor = ColumnTransformer([  
 ('nom\_x', OneHotEncoder(drop='first'), ['nom\_x'])  
], remainder='passthrough')  
  
model = make\_pipeline(preprocessor, LinearRegression())  
model.fit(X, y)  
predictions = model.predict(X)  
  
print("Coefficients for each predictor:", model.named\_steps['linearregression'].coef\_)

**Interpretation:** - One-hot encoding allows inclusion of nominal variables in regression. - Coefficients show the effect of each category relative to the reference.

#### Logistic Regression

from sklearn.linear\_model import LogisticRegression  
from statsmodels.stats.outliers\_influence import variance\_inflation\_factor  
from sklearn.metrics import roc\_auc\_score  
import numpy as np  
  
# Use continuous and ordinal predictors, binary outcome from df  
X = df[['cont\_x', 'ord\_x']].values  
y = df['binary\_y'].values  
  
# Check for multicollinearity (VIF)  
vif = [variance\_inflation\_factor(X, i) for i in range(X.shape[1])]  
print('VIF:', vif)  
  
# Fit model  
model = LogisticRegression()  
model.fit(X, y)  
predictions = model.predict(X)  
probs = model.predict\_proba(X)[:, 1]  
roc\_auc = roc\_auc\_score(y, probs)  
print(f"ROC-AUC: {roc\_auc:.3f}")

**Interpretation:** - VIF > 5 indicates problematic multicollinearity. - ROC-AUC > 0.7 suggests good classification performance.

#### Ordinal Regression (Ordinal Logistic Regression)

import statsmodels.api as sm  
from statsmodels.miscmodels.ordinal\_model import OrderedModel  
  
# Use ordinal outcome and continuous predictor  
X = df[['cont\_x']]  
y = df['ordinal\_y']  
  
# Fit ordinal logistic regression  
model = OrderedModel(y, sm.add\_constant(X), distr='logit')  
result = model.fit(method='bfgs')  
print(result.summary())

**Interpretation:** - OrderedModel is used for ordinal outcomes. - Coefficients indicate the effect of predictors on the odds of being in a higher category.

#### Polynomial Regression

from sklearn.preprocessing import PolynomialFeatures  
from sklearn.linear\_model import LinearRegression  
from sklearn.pipeline import make\_pipeline  
import matplotlib.pyplot as plt  
  
# Use continuous independent and dependent variables from df  
X = df[['cont\_x']].values  
y = df['cont\_y'].values  
  
# Visual check for non-linearity  
plt.scatter(X, y)  
plt.title('Non-linearity Check')  
plt.xlabel('cont\_x')  
plt.ylabel('cont\_y')  
plt.show()  
  
# Fit polynomial regression (degree=2)  
model = make\_pipeline(PolynomialFeatures(degree=2), LinearRegression())  
model.fit(X, y)  
predictions = model.predict(X)  
  
print("First 5 predictions:", predictions[:5])

**Interpretation:** - Polynomial regression captures non-linear relationships. - Risk of overfitting increases with higher-degree polynomials.

#### Ridge and Lasso Regression

from sklearn.linear\_model import Ridge, Lasso  
  
# Use continuous independent and dependent variables from df  
X = df[['cont\_x']].values  
y = df['cont\_y'].values  
  
# Fit Ridge regression  
ridge\_model = Ridge(alpha=1.0)  
ridge\_model.fit(X, y)  
print("Ridge coefficients:", ridge\_model.coef\_)  
  
# Fit Lasso regression  
lasso\_model = Lasso(alpha=0.1)  
lasso\_model.fit(X, y)  
print("Lasso coefficients:", lasso\_model.coef\_)

**Interpretation:** - Ridge and Lasso add regularization to reduce overfitting and handle multicollinearity. - Lasso can shrink some coefficients to zero (variable selection).

#### Quantile Regression

import statsmodels.api as sm  
  
# Use continuous independent and dependent variables from df  
X = sm.add\_constant(df['cont\_x'])  
y = df['cont\_y']  
quantile\_model = sm.QuantReg(y, X).fit(q=0.5)  
print(quantile\_model.summary())

**Interpretation:** - Quantile regression estimates conditional medians or other quantiles, robust to outliers.

#### Poisson Regression with Categorical Predictor

import statsmodels.api as sm  
import pandas as pd  
  
# One-hot encode nominal predictor  
X = pd.get\_dummies(df['nom\_x'], drop\_first=True)  
X = sm.add\_constant(X)  
y = df['count\_y']  
poisson\_model = sm.GLM(y, X, family=sm.families.Poisson()).fit()  
print(poisson\_model.summary())

**Interpretation:** - Poisson regression can use categorical predictors (after encoding) to model count outcomes.

#### Multinomial Logistic Regression (Nominal Outcome)

import statsmodels.api as sm  
import pandas as pd  
  
# Use continuous predictor and nominal outcome  
y = df['nominal\_y'].astype('category').cat.codes  
X = sm.add\_constant(df[['cont\_x']])  
mnlogit\_model = sm.MNLogit(y, X).fit()  
print(mnlogit\_model.summary())

**Interpretation:** - Multinomial logistic regression models nominal outcomes with more than two categories. - Coefficients represent log-odds of each category relative to the reference.

#### Narrative Workflow: Linear Regression Example

This section walks through a typical linear regression analysis using the generated dataset, modeling the full workflow from data exploration to diagnostics and interpretation.

import matplotlib.pyplot as plt  
import seaborn as sns  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error, r2\_score  
from scipy.stats import shapiro  
from statsmodels.stats.diagnostic import het\_breuschpagan  
import statsmodels.api as sm  
  
# 1. Data exploration  
sns.histplot(df['cont\_x'], kde=True)  
plt.title('Distribution of cont\_x')  
plt.show()  
sns.scatterplot(x='cont\_x', y='cont\_y', data=df)  
plt.title('cont\_x vs. cont\_y')  
plt.show()  
  
# 2. Model fitting  
X = df[['cont\_x']].values  
y = df['cont\_y'].values  
model = LinearRegression()  
model.fit(X, y)  
predictions = model.predict(X)  
  
# 3. Diagnostics  
residuals = y - predictions  
mse = mean\_squared\_error(y, predictions)  
r2 = r2\_score(y, predictions)  
print(f"MSE: {mse:.2f}, R^2: {r2:.2f}")  
  
# Residual analysis  
plt.scatter(predictions, residuals)  
plt.axhline(0, color='red', linestyle='--')  
plt.title('Residuals vs. Fitted')  
plt.xlabel('Fitted values')  
plt.ylabel('Residuals')  
plt.show()  
  
# Normality test  
stat, p = shapiro(residuals)  
print(f"Shapiro-Wilk p-value: {p:.3f}")  
  
# Heteroscedasticity test  
bp\_test = het\_breuschpagan(residuals, sm.add\_constant(X))  
print(f"Breusch-Pagan p-value: {bp\_test[1]:.3f}")

**Interpretation:** - Data exploration reveals the distribution and relationship between variables. - Model fit is assessed by MSE and R². - Residual plots and tests check for normality and homoscedasticity. - If assumptions are violated, consider transformations or robust regression.