REGRESSION MODEL CHEAT SHEET

◆ 1. What is Regression?

Regression is a supervised learning method used to model the relationship between a **dependent variable** and one or more **independent variables**.

◆ 2. Types of Regression

Type	Description
Linear Regression	Predicts continuous outcome with linear terms
Multiple Regression	More than one predictor variable
Polynomial Regression	Includes non-linear (squared, cubic) terms
Logistic Regression	Classification model using logistic function
Ridge/Lasso	Regularized regression to prevent overfitting
ElasticNet	Combines Ridge and Lasso penalties

◆ 3. Simple Linear Regression

Model:

Y=β0+β1X+εY = \beta_0 + \beta_1X + \varepsilonY=β0+β1X+ε

Goal: Find the line that minimizes residuals (errors).

Cost Function (MSE):

MSE=1n∑i=1n(yi−y^i)2\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2MSE=n1 i=1∑n(yi−y^i)2

◆ 4. Multiple Linear Regression

Model:

Y=β0+β1X1+β2X2+⋯+βnXn+εY = \beta_0 + \beta_1X_1 + \beta_2X_2 + \cdots + \beta_nX_n + \varepsilonY=β0+β1X1+β2X2+⋯+βnXn+ε

---

5. Key Metrics

Metric	Meaning
R^2	% of variance explained by the model
Adjusted R^2	Adjusted for number of predictors
RMSE	Root mean square error
MAE	Mean absolute error
p-values	Test significance of coefficients
VIF	Detect multicollinearity

---

6. Assumptions of Linear Regression

1. **Linearity**
2. **Independence of errors**
3. **Homoscedasticity (constant variance of residuals)**
4. **Normality of residuals**
5. **No multicollinearity**

---

7. Python Code Snippet

```
python
CopyEdit
import pandas as pd
import numpy as np
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Load Data
df = pd.read_csv("data.csv")

# Define variables
X = df[['X1', 'X2']]  # independent variables
y = df['Y']           # dependent variable

# Add constant for statsmodels
X_sm = sm.add_constant(X)

# Fit model (statsmodels)
model = sm.OLS(y, X_sm).fit()
print(model.summary())

# OR using sklearn
lr = LinearRegression()
lr.fit(X, y)
```

```
preds = lr.predict(X)

print("R²:", r2_score(y, preds))
print("RMSE:", np.sqrt(mean_squared_error(y, preds)))
```

🔹 8. Plotting Diagnostic Charts

```
python
CopyEdit
import matplotlib.pyplot as plt
import seaborn as sns

# Residuals plot
residuals = y - preds
sns.residplot(x=preds, y=residuals, lowess=True)
plt.xlabel("Predicted")
plt.ylabel("Residuals")
plt.title("Residual Plot")
plt.show()
```

🔹 9. When to Use Regularization

- Many predictors ($p > n$)
- High multicollinearity
- Overfitting risk

Lasso (L1):

Loss+λ∑|β|\text{Loss} + \lambda \sum |\beta|Loss+λ∑|β|

Ridge (L2):

Loss+λ∑β2\text{Loss} + \lambda \sum \beta^2Loss+λ∑β2