Q1. What is Elastic Net Regression and how does it differ from other regression techniques?

Elastic Net Regression is a linear regression technique that combines the penalties of **Lasso (L1L_1L1)** and **Ridge (L2L_2L2)** regression:

Cost Function: RSSEN=∑i=1n(yi−y^i)2+λ1∑j=1p∣βj∣+λ2∑j=1pβj2\text{Cost Function: } \text{RSS}_{EN} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} \beta_j^2Cost Function: RSSEN=i=1∑n(yi−y^i)2+λ1j=1∑p∣βj∣+λ2j=1∑pβj2

Or, in the more common form used in libraries:

RSSEN=∑i=1n(yi−y^i)2+λ(α∑j=1p∣βj∣+1−α2∑j=1pβj2)\text{RSS}_{EN} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \Big(\alpha \sum_{j=1}^{p} |\beta_j| + \frac{1-\alpha}{2} \sum_{j=1}^{p} \beta_j^2 \Big)RSSEN=i=1∑n(yi−y^i)2+λ(αj=1∑p∣βj∣+21−αj=1∑pβj2)

Where:

- λ≥0\lambda \ge 0λ≥0 controls the overall strength of regularization.
- α∈[0,1]\alpha \in [0,1]α∈[0,1] balances between **Lasso (α=1\alpha=1α=1)** and **Ridge (α=0\alpha=0α=0)**.

Differences from other regression techniques:

Regression Type	Penalty	Feature Selection	Multicollinearity Handling
OLS	None	No	Poor
Ridge	L2L_2L2	No	Good
Lasso	L1L_1L1	Yes	Partially
Elastic Net	L1+L2L_1 + L_2L1+L2	Yes	Good

Elastic Net **combines the advantages of Lasso (feature selection) and Ridge (stability with correlated features)**.

Q2. How do you choose the optimal values of the regularization parameters for Elastic Net Regression?

Two parameters need tuning:

1. **λ\lambdaλ** – overall regularization strength.
2. **α\alphaα** – mix ratio between Lasso (L1L_1L1) and Ridge (L2L_2L2).

Selection methods:

- **Cross-Validation (CV)**:
  - Use **GridSearchCV** or **ElasticNetCV** in Python to test combinations of (λ,α)(\lambda, \alpha)(λ,α) and pick the pair with minimum validation error.
- **Path plots**: Visualize how coefficients change with λ\lambdaλ.

Rule of thumb:

- α≈0.5\alpha \approx 0.5α≈0.5 is a good starting point, then tune via CV.

Q3. What are the advantages and disadvantages of Elastic Net Regression?

Advantages:

1. Performs **automatic feature selection** like Lasso.
2. Handles **highly correlated features** better than Lasso.
3. Reduces overfitting with **regularization**.
4. Combines benefits of Ridge and Lasso in one model.

Disadvantages:

1. Requires tuning **two hyperparameters** (λ\lambdaλ and α\alphaα), which can be computationally intensive.

2. Coefficients are **biased** due to shrinkage.
3. May still arbitrarily select among correlated features depending on α\alphaα.

Q4. What are some common use cases for Elastic Net Regression?

- **Genomics**: Selecting predictive genes from thousands of correlated genes.
- **Finance**: Predicting stock returns from highly correlated financial indicators.
- **Marketing analytics**: Modeling customer behavior with many correlated demographic or usage features.
- **High-dimensional datasets**: Where p>np > np>n (more features than observations) and feature selection + stability is needed.

Q5. How do you interpret the coefficients in Elastic Net Regression?

- **Non-zero coefficients** indicate important predictors.
- **Zero coefficients** (if any) indicate excluded features.
- **Magnitude of coefficients** indicates relative influence (after standardization).
- Coefficients are **biased** due to regularization; interpret cautiously for causal claims.

Tip: Standardize features before modeling for meaningful coefficient comparisons.

Q6. How do you handle missing values when using Elastic Net Regression?

- **Elastic Net cannot handle NaNs directly**. Options:
    1. **Imputation**: Fill missing values with mean, median, or use KNN/Iterative imputer.
    2. **Remove rows** with missing data (if small proportion).

3. **Pipeline**: Use `sklearn.pipeline.Pipeline` with `SimpleImputer` + `ElasticNet`.

Q7. How do you use Elastic Net Regression for feature selection?

- Set α>0\alpha > 0α>0 so the **Lasso component is active**.
- Fit the model → coefficients that are **exactly zero** indicate unimportant features.
- Non-zero coefficients indicate **selected features**.
- Advantage over Lasso: it **selects correlated features together** instead of arbitrarily dropping some.

Q8. How do you pickle and unpickle a trained Elastic Net Regression model in Python?

Pickle a model:

```
import pickle
from sklearn.linear_model import ElasticNet

# Train model
model = ElasticNet(alpha=0.5, l1_ratio=0.7)
model.fit(X_train, y_train)

# Save to file
with open("elastic_net_model.pkl", "wb") as f:
    pickle.dump(model, f)
```

Unpickle a model:

```
# Load model from file
with open("elastic_net_model.pkl", "rb") as f:
    loaded_model = pickle.load(f)
```

```
# Use model
predictions = loaded_model.predict(X_test)
```

Q9. What is the purpose of pickling a model in machine learning?

- **Persistence**: Save trained models to disk for later use without retraining.
- **Deployment**: Load models into production systems.
- **Reproducibility**: Share or reuse models with the same parameters and training state.
- **Efficiency**: Avoid expensive retraining on large datasets.
