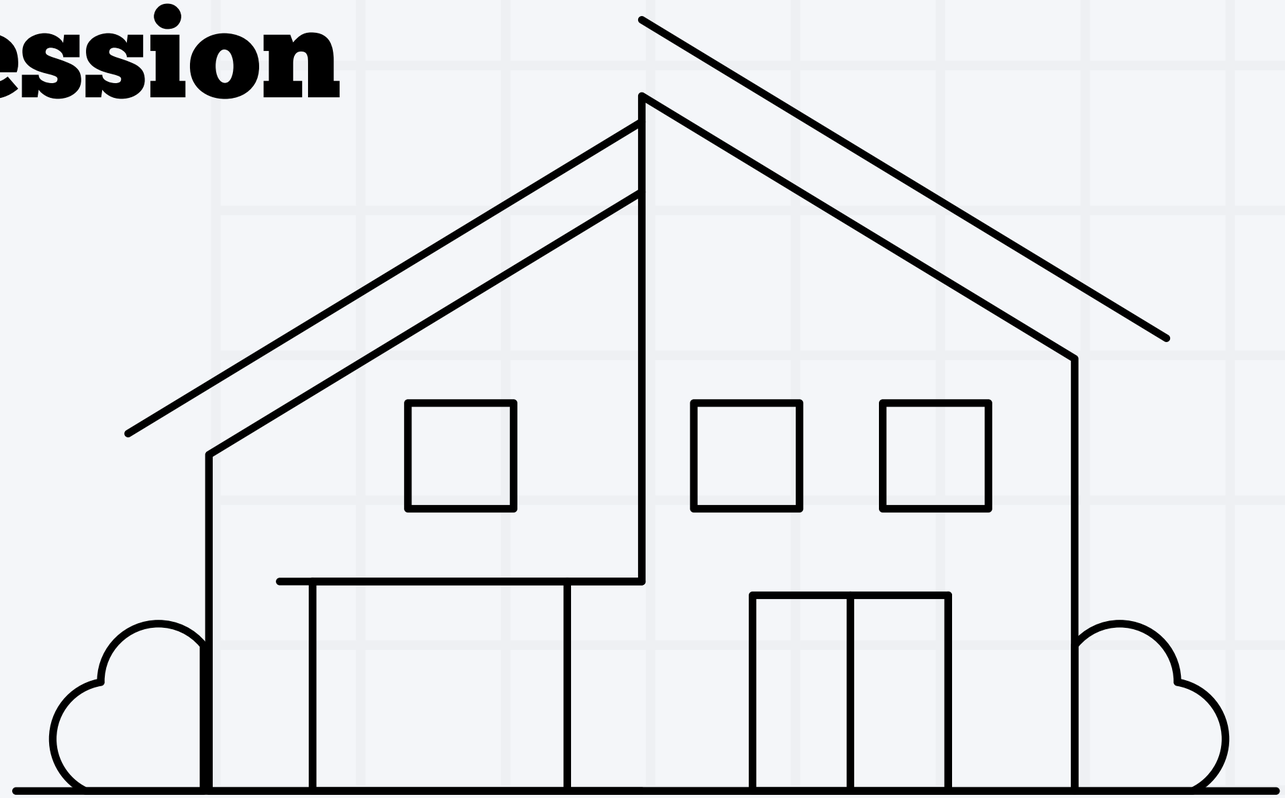
Advanced Regression Techniques

Group 32



Group members:
A121620 范凱捷
110550143 洪巧芸
110511081 洪峻德
111705015 張詠翔
111705021 黃意庭

Outline

01 Ames Housing Dataset

02 Data Preprocessing

03 Math of Evaluation

04 SKLearn

05 XGBoost

06 Evaluation

/01

AMES HOUSING DATASET

What does it contain ?

The Ames Housing dataset contains 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa with the goal of predicting the selling price.

Why is it so well-known?

It's an incredible alternative for data scientists looking for a modernized and expanded version of the often cited Boston Housing dataset.

Where is it use for?

It’s a widely used dataset for learning fundamentals of regression analysis while providing good practice cleaning and exploring a dataset.

/02

Data Preprocessing

Step 0 Data Exploration

Take a look at the dataset

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	...
count	1460.000000	1460.000000	1201.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1452.000000	1460.000000	...
mean	730.500000	56.897260	70.049958	10516.828082	6.099315	5.575342	1971.267808	1984.865753	103.685262	443.639726	...
std	421.610009	42.300571	24.284752	9981.264932	1.382997	1.112799	30.202904	20.645407	181.066207	456.098091	...
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1872.000000	1950.000000	0.000000	0.000000	...
25%	365.750000	20.000000	59.000000	7553.500000	5.000000	5.000000	1954.000000	1967.000000	0.000000	0.000000	...
50%	730.500000	50.000000	69.000000	9478.500000	6.000000	5.000000	1973.000000	1994.000000	0.000000	383.500000	...
75%	1095.250000	70.000000	80.000000	11601.500000	7.000000	6.000000	2000.000000	2004.000000	166.000000	712.250000	...
max	1460.000000	190.000000	313.000000	215245.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.000000	...

8 rows × 38 columns

Step 1 Data Encoding

Encode string categorical data into numerical values in both training and testing dataset.

Step 2 Handling of Missing Data

- Drop NaN for training dataset
- Fill NaN for testing dataset
- Drop “ID” column of the dataset.

Step 3 Feature Transformation

Scale the data using standardisation

/02

Data Preprocessing

Step 4 Feature Selection

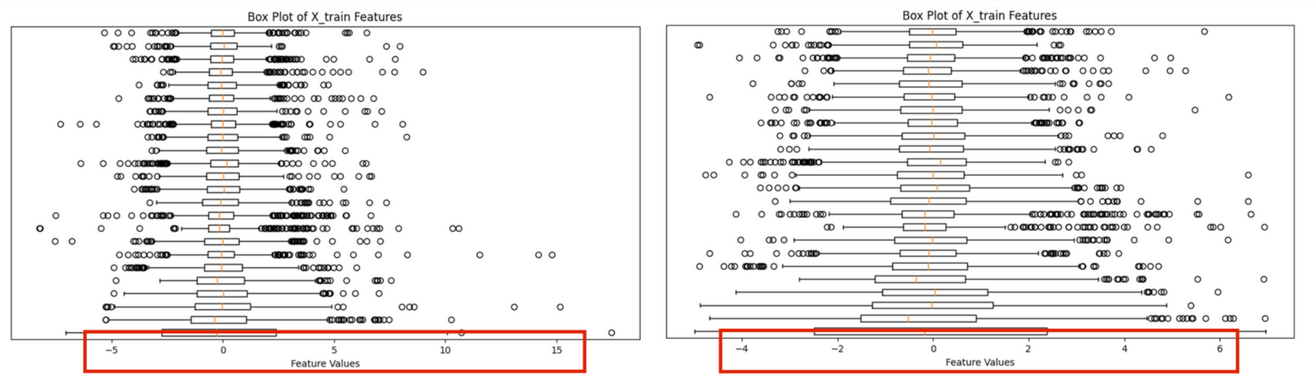
Principal Component Analysis (PCA) was used to filter out the features that contain most of the information

Step 5 Data Spilt

- train_test_split function was used to split the training dataset into training and validation sets
- Validation set used to test the performance of the model

Step 6 Outlier Removal

Remove datapoints that are far away from the majority of the datapoints.



/03

MEAN SQUARE ERROR

It’s used to measure the prediction accuracy of a model. The lower the value for MSE, the better a model is able to predict values accurately.

MSE = (1/n) * Σ(actual – prediction)^2

- **n** – sample size
- **actual** – the actual data value
- **prediction** – the predicted data value

/03

ROOT MEAN SQUARE ERROR

It’s used to assess model accuracy.
The lower the value for RMS, the better a model is able to predict values accurately.

RMS

= [(1/n) * Σ(prediction – actual)^2]^(1/2)

- **n** – sample size
- **actual** – the actual data value
- **prediction** – the predicted data value

/03

R-SQUARED

It’s a measure that provides information about the goodness of fit of a model. It will take values between 0 and 1. The higher value for R-square, the better our model is able to predict values accurately.

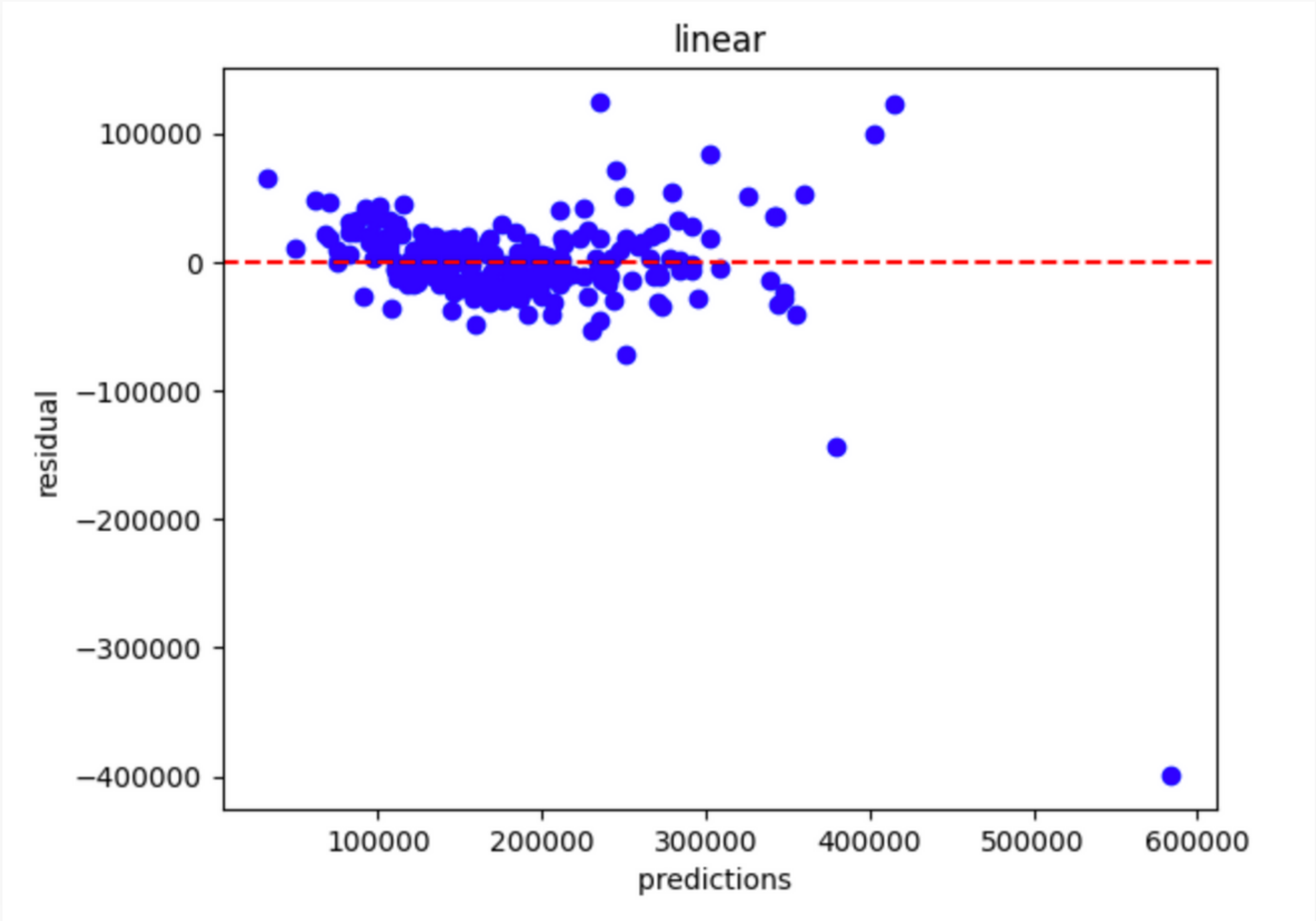
r =

n (∑xy) – ∑x ∑y / {[n* (∑x^2– (∑x)^2)] * [n* (∑y^2 – (∑y)^2)]}^(1/2)

- **n** – sample size
- **x** – the predicted data value
- **y** – the actual data value

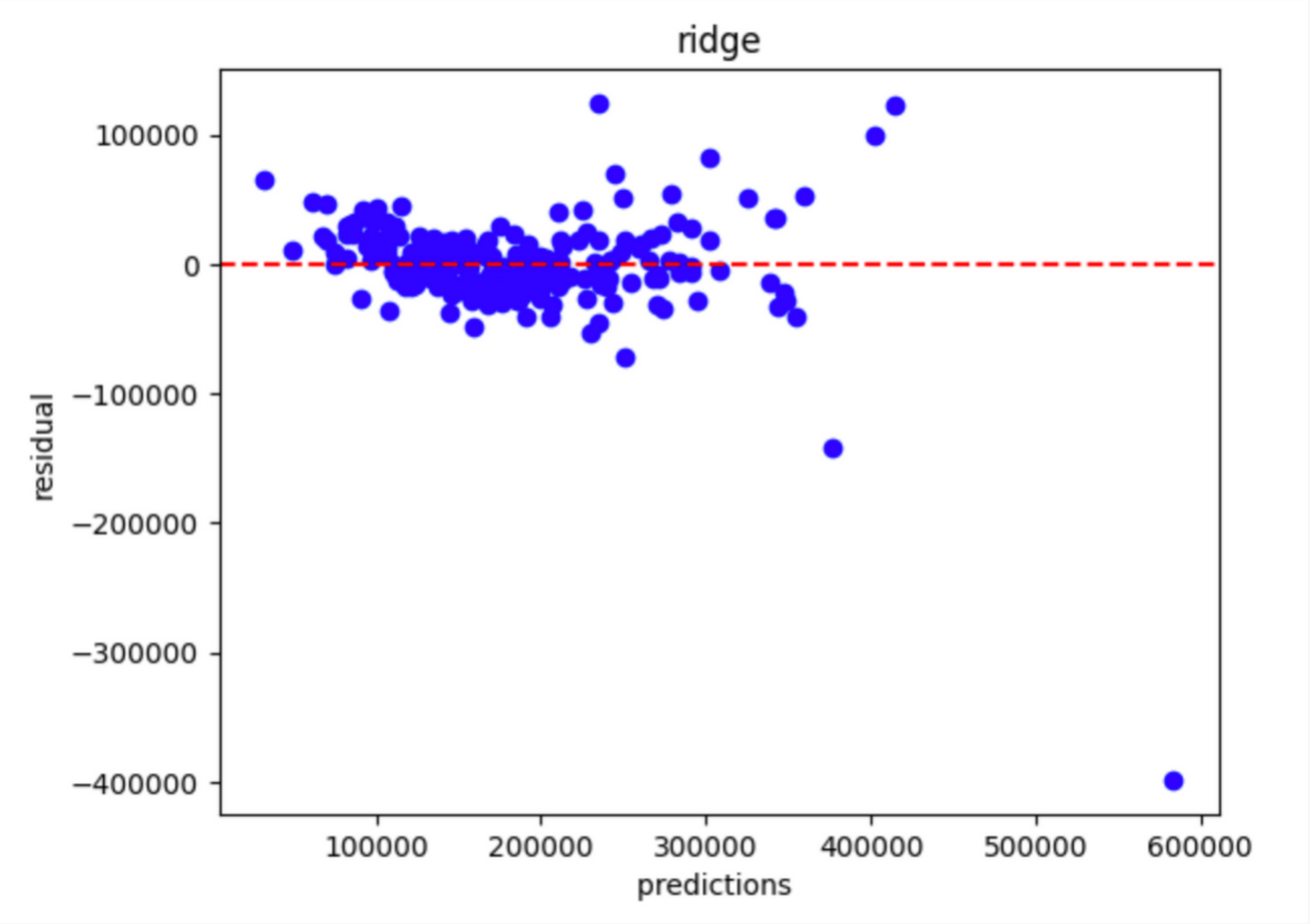
/04 SKLEARN

```
X = np.array(X_train)
y = np.array(y_train)

model = LinearRegression()

model.fit(X, y)

coef = model.coef_
intercept = model.intercept_

new_data = np.array(X_test)
predictions = model.predict(new_data)
residuals = y_test - predictions

plt.scatter(predictions, residuals, color='blue')
plt.axhline(y=0, color='red', linestyle='--')
plt.xlabel('predictions ')
plt.ylabel('residual')
plt.title('linear')
plt.show()
```



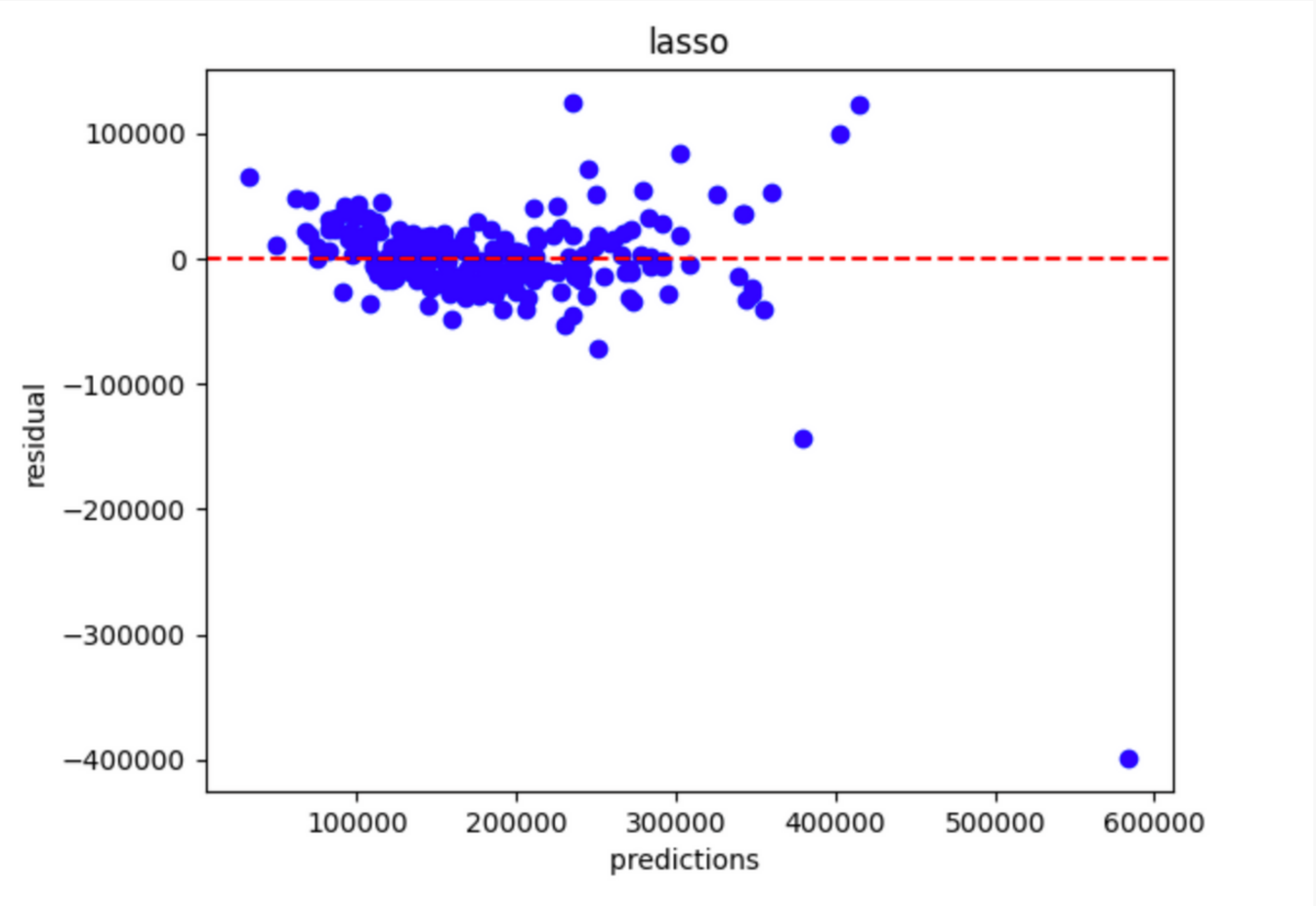
/04 SKLEARN

```python
from sklearn.linear_model import Ridge
X = np.array(X_train)
y = np.array(y_train)

ridge_model = Ridge(alpha=1.0)

ridge_model.fit(X, y)

coef = ridge_model.coef_
intercept = ridge_model.intercept_

new_data = np.array(X_test)
predictions = ridge_model.predict(new_data)

residuals = y_test - predictions

plt.scatter(predictions, residuals, color='blue')
plt.axhline(y=0, color='red', linestyle='--')
plt.xlabel('predictions ')
plt.ylabel('residual')
plt.title('ridge')
plt.show()
```



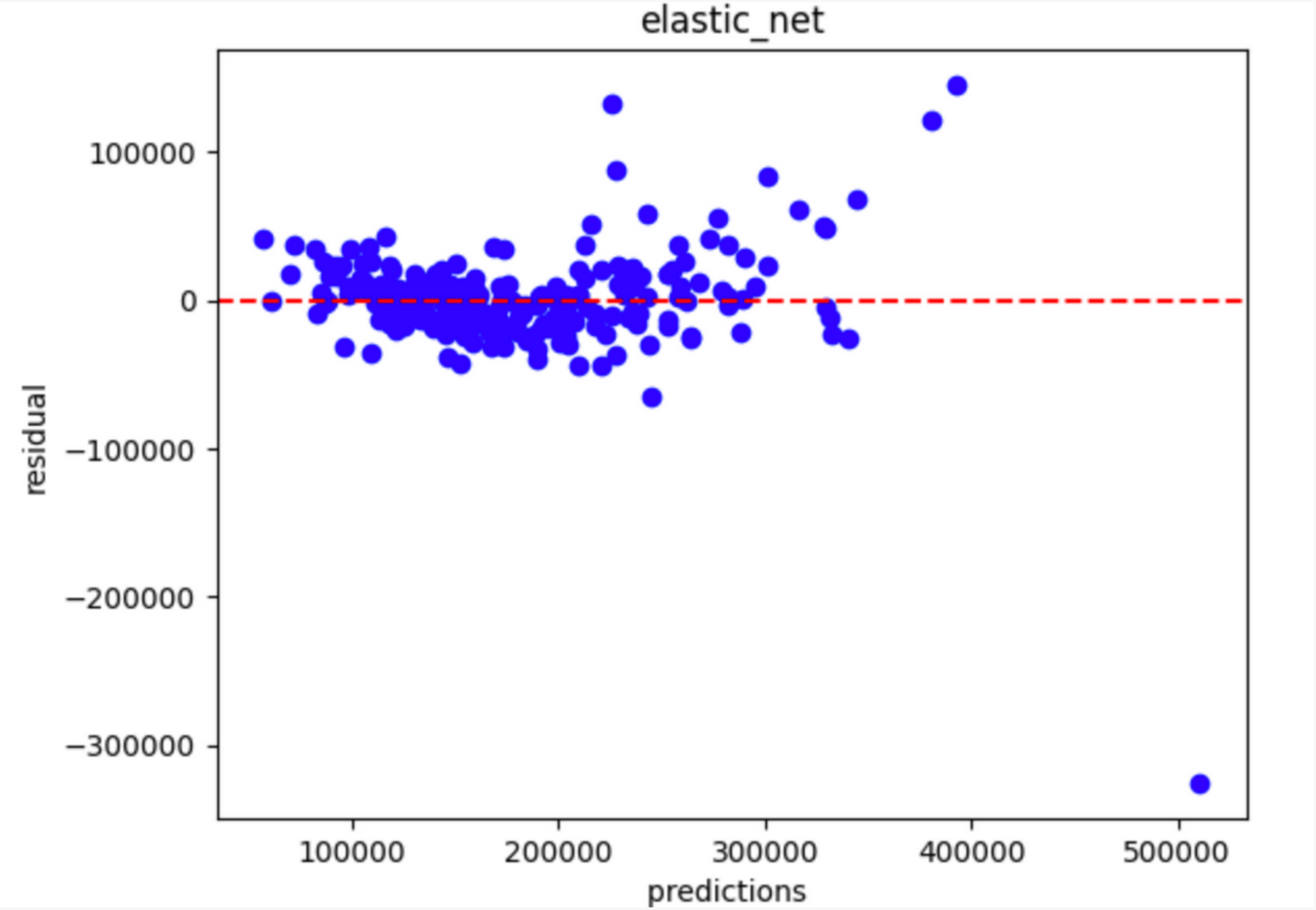
/04 SKLEARN

```python
from sklearn.linear_model import Lasso

X = np.array(X_train)
y = np.array(y_train)

lasso_model = Lasso(alpha=1.0)

lasso_model.fit(X, y)

coef = lasso_model.coef_
intercept = lasso_model.intercept_

new_data = np.array(X_test)
predictions = lasso_model.predict(new_data)

residuals = y_test - predictions

plt.scatter(predictions, residuals, color='blue')
plt.axhline(y=0, color='red', linestyle='--')
plt.xlabel('predictions ')
plt.ylabel('residual')
plt.title('lasso')
plt.show()
```



/04 SKLEARN

```python
from sklearn.linear_model import ElasticNet
import matplotlib.pyplot as plt

X = np.array(X_train)
y = np.array(y_train)

elastic_net_model = ElasticNet(alpha=1.0, l1_ratio=0.5)

elastic_net_model.fit(X, y)

coef = elastic_net_model.coef_
intercept = elastic_net_model.intercept_

new_data = np.array(X_test)
predictions = elastic_net_model.predict(new_data)

residuals = y_test - predictions

plt.scatter(predictions, residuals, color='blue')
plt.axhline(y=0, color='red', linestyle='--')
plt.xlabel('predictions ')
plt.ylabel('residual')
plt.title('elastic_net')
plt.show()
```



/05 XGBOOST

```
#XGBoost
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, accuracy_score

#For regression
regressor = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=2000,
                             learning_rate=0.01, max_depth=3, random_state=123)

regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
y_pred_pca = regressor.predict(X_test_pca)
print(y_pred_pca.shape)
```

/06 EVALUATION

```
mse = mean_squared_error(y_test, y_pred, squared=False)
rmse = math.sqrt(mse)
print(f"Mean Squared Error: {mse}")
print(f"Root Mean Squared Error: {rmse}")

r2 = r2_score(y_test, y_pred)

print(f"R-squared (R^2): {r2}")
```

```
Mean Squared Error: 30894.40463829525
Root Mean Squared Error: 175.76804214161132
R-squared (R^2): 0.8351575214943456
```

Thank You