

Nama: Rizki Aprilia Rahman


NIM: 1103213007

Kelas: TK-45-G04


TUGAS 1 PEMBELAJARAN MESIN

Membuat model classification dengan logistics regression dan K-NN

- Iris.csv

 Iris.csv
Spreadsheet

buatkan model classification dengan logistics regression dan K-NN dalam codingan google colab dari file tersebut

 Dataset ini berisi fitur-fitur dari dataset Iris yang mencakup panjang dan lebar sepal, panjang dan lebar petal, serta kolom target yang menunjukkan spesies bunga.

Sekarang, saya akan membuat model klasifikasi menggunakan **Logistic Regression** dan **K-NN** berdasarkan file ini.

Langkah-Langkah:

1. Menghapus kolom yang tidak diperlukan (misalnya **Id**).
2. Mengubah kolom target (**Species**) menjadi data numerik.
3. Membagi dataset menjadi data training dan testing.
4. Melatih model Logistic Regression dan K-NN.
5. Mengevaluasi akurasi model.

Saya akan menulis kodenya sekarang. ↩

Hasil dari model klasifikasi menggunakan **Logistic Regression** dan **K-NN** menunjukkan akurasi yang sangat baik: ↓

```
# Let's read the uploaded CSV file and inspect its contents to prepare it for classification using Logistic Regression
import pandas as pd

# Load the dataset from the uploaded file
file_path = 'Iris.csv'
df = pd.read_csv(file_path)

# Display the first few rows of the dataset
df.head(10)
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa

```
06 from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report

# Step 1: Remove unnecessary 'Id' column and encode target labels
df = df.drop(columns=['Id'])
label_encoder = LabelEncoder()
df['Species'] = label_encoder.fit_transform(df['Species'])

# Step 2: Define features (X) and target (y)
X = df.drop(columns=['Species']) # Features
y = df['Species'] # Target

# Step 3: Split the dataset into training and test sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 4: Standardize the data (important for K-NN)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Step 5: Train Logistic Regression model
logreg = LogisticRegression(max_iter=200)
logreg.fit(X_train, y_train)

# Step 6: Train K-Nearest Neighbors (K-NN) model
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

06 # Step 7: Make predictions with both models
y_pred_logreg = logreg.predict(X_test)
y_pred_knn = knn.predict(X_test)

# Step 8: Evaluate Logistic Regression model
logreg_accuracy = accuracy_score(y_test, y_pred_logreg)
logreg_report = classification_report(y_test, y_pred_logreg)

# Step 9: Evaluate K-NN model
knn_accuracy = accuracy_score(y_test, y_pred_knn)
knn_report = classification_report(y_test, y_pred_knn)

# Menampilkan hasil secara rapi
print(f"Logistic Regression Accuracy: {logreg_accuracy:.2f}")
print("Logistic Regression Classification Report:")
print(logreg_report)

print(f"K-Nearest Neighbors (K-NN) Accuracy: {knn_accuracy:.2f}")
print("K-Nearest Neighbors (K-NN) Classification Report:")
print(knn_report)
```

```
Logistic Regression Accuracy: 1.00
Logistic Regression Classification Report:
      precision    recall  f1-score   support

0             1.00      1.00      1.00        10
1             1.00      1.00      1.00         9
2             1.00      1.00      1.00        11


 accuracy          1.00
macro avg          1.00
weighted avg       1.00

K-Nearest Neighbors (K-NN) Accuracy: 1.00
K-Nearest Neighbors (K-NN) Classification Report:
      precision    recall  f1-score   support

0             1.00      1.00      1.00        10
1             1.00      1.00      1.00         9
2             1.00      1.00      1.00        11

 accuracy          1.00
macro avg          1.00
weighted avg       1.00
```

- Social Media ads

 Social_Network_Ads.csv
Spreadsheet

buatkan model classification dengan logistics regression dan K-NN dalam codingan google colab dari file tersebut dalam bahasa Indonesia

Berikut adalah kode untuk melakukan klasifikasi menggunakan Logistic Regression dan K-Nearest Neighbors (K-NN) pada dataset yang telah diunggah menggunakan Google Colab, dalam bahasa Indonesia.

```
python Copy code

# Import pustaka yang diperlukan
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Load dataset
file_path = '/path/to/your/Social_Network_Ads.csv' # Ganti dengan path file yang sesuai
df = pd.read_csv(file_path)

# Melihat beberapa baris pertama dari data
print(df.head())
```

```
# Let's read the uploaded CSV file and inspect its contents to prepare it for classification
import pandas as pd

# Load the dataset from the uploaded file
file_path = 'Social_Network_Ads.csv'
df = pd.read_csv(file_path)

# Display the first few rows of the dataset
df.head(10)

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report

# Step 1: Define features (X) and target (y)
X = df[['Age', 'EstimatedSalary']] # Features
y = df['Purchased'] # Target

# Step 2: Split the dataset into training and test sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 3: Standardize the data (important for K-NN)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Step 4: Train Logistic Regression model
logreg = LogisticRegression(max_iter=200)
logreg.fit(X_train, y_train)

# Step 5: Train K-Nearest Neighbors (K-NN) model
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

# Step 6: Make predictions with both models
y_pred_logreg = logreg.predict(X_test)
y_pred_knn = knn.predict(X_test)

# Step 7: Evaluate Logistic Regression model
logreg_accuracy = accuracy_score(y_test, y_pred_logreg)
logreg_report = classification_report(y_test, y_pred_logreg)

# Step 8: Evaluate K-NN model
knn_accuracy = accuracy_score(y_test, y_pred_knn)
knn_report = classification_report(y_test, y_pred_knn)

# Menampilkan hasil secara rapi
print(f"Logistic Regression Accuracy: {logreg_accuracy:.2f}")
print("Logistic Regression Classification Report:")
print(logreg_report)

print(f"K-Nearest Neighbors (K-NN) Accuracy: {knn_accuracy:.2f}")
print("K-Nearest Neighbors (K-NN) Classification Report:")
print(knn_report)
```

Logistic Regression Accuracy: 0.86

Logistic Regression Classification Report:

	precision	recall	f1-score	support
0	0.85	0.96	0.90	52
1	0.90	0.68	0.78	28
accuracy			0.86	80
macro avg	0.88	0.82	0.84	80
weighted avg	0.87	0.86	0.86	80

K-Nearest Neighbors (K-NN) Accuracy: 0.91

K-Nearest Neighbors (K-NN) Classification Report:

	precision	recall	f1-score	support
0	0.94	0.92	0.93	52
1	0.86	0.89	0.88	28
accuracy			0.91	80
macro avg	0.90	0.91	0.90	80
weighted avg	0.91	0.91	0.91	80

Regression model: Membuat 3 model regression (Simple Linear Regression, Multiple Linear Regression, dan Polynomial Regression)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score

# Load dataset
data = pd.read_csv('Salary_Data.csv')

# Melihat 5 baris pertama dari dataset
data.head()

# Memisahkan fitur dan target
X = data[['YearsExperience']] # Fitur
y = data['Salary'] # Target

# Membagi data menjadi set pelatihan dan pengujian
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 1. Simple Linear Regression
# Membuat dan melatih model regresi linear sederhana
linear_regressor = LinearRegression()
linear_regressor.fit(X_train, y_train)

# Prediksi menggunakan model regresi linear
y_pred_linear = linear_regressor.predict(X_test)

# Menampilkan hasil regresi linear
print("Simple Linear Regression:")
print("Mean Squared Error (MSE):", mean_squared_error(y_test, y_pred_linear))
print("R-squared:", r2_score(y_test, y_pred_linear))

# Visualisasi regresi linear sederhana
plt.scatter(X_test, y_test, color='blue')
plt.plot(X_test, y_pred_linear, color='red')
plt.title("Regresi Linear Sederhana")
plt.xlabel("Tahun Pengalaman")
plt.ylabel("Gaji")
plt.show()

# 2. Multiple Linear Regression (untuk contoh ini, karena hanya ada satu fitur)

# 3. Polynomial Regression
# Membuat fitur polinomial
poly_features = PolynomialFeatures(degree=3) # Degree polinomial dapat diubah
X_poly_train = poly_features.fit_transform(X_train)
X_poly_test = poly_features.transform(X_test)

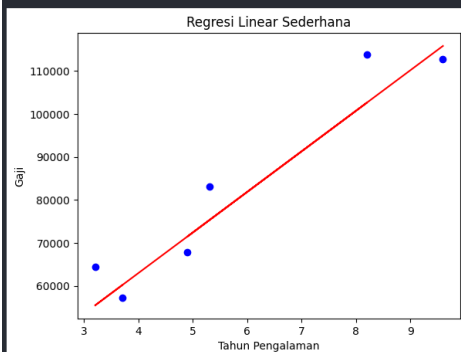
# Membuat dan melatih model regresi polinomial
poly_regressor = LinearRegression()
poly_regressor.fit(X_poly_train, y_train)

# Prediksi menggunakan model regresi polinomial
y_pred_poly = poly_regressor.predict(X_poly_test)

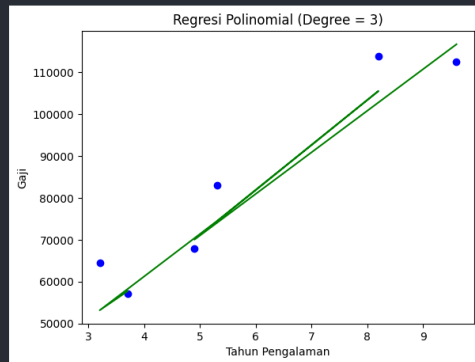
# Menampilkan hasil regresi polinomial
print("Polynomial Regression:")
print("Mean Squared Error (MSE):", mean_squared_error(y_test, y_pred_poly))
print("R-squared:", r2_score(y_test, y_pred_poly))

# Visualisasi regresi polinomial
plt.scatter(X_test, y_test, color='blue')
plt.plot(X_test, y_pred_poly, color='green')
plt.title("Regresi Polinomial (Degree = 3)")
plt.xlabel("Tahun Pengalaman")
plt.ylabel("Gaji")
plt.show()
```

Simple Linear Regression:
Mean Squared Error (MSE): 49830096.85590839
R-squared: 0.9024461774180497



Polynomial Regression:
Mean Squared Error (MSE): 48605815.33188739
R-squared: 0.9048429887854863



```
# Import library yang dibutuhkan
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error, r2_score

# Load dataset
data = pd.read_csv('50_Startups.csv')

# Mengonversi variabel kategorikal (State) menjadi variabel dummy
data = pd.get_dummies(data, drop_first=True)

# Memisahkan fitur dan target untuk simple linear regression
X_simple = data[['R&D Spend']] # Fitur sederhana
y = data['Profit'] # target

# Memisahkan fitur dan target untuk multiple regression
X_multiple = data[['R&D Spend', 'Administration', 'Marketing Spend']] # Fitur ganda

# Membagi data menjadi set pelatihan dan pengujian
X_train_simple, X_test_simple, y_train, y_test = train_test_split(X_simple, y, test_size=0.2, random_state=42)
X_train_multiple, X_test_multiple, y_train, y_test = train_test_split(X_multiple, y, test_size=0.2, random_state=42)

# 1. Simple Linear Regression
# Membuat dan melatih model regresi linear sederhana
linear_regressor = LinearRegression()
linear_regressor.fit(X_train_simple, y_train)

# Prediksi menggunakan model regresi linear sederhana
y_pred_linear = linear_regressor.predict(X_test_simple)

# Menampilkan hasil regresi linear sederhana
print("Simple Linear Regression:")
print("Mean Squared Error (MSE):", mean_squared_error(y_test, y_pred_linear))
print("R-squared:", r2_score(y_test, y_pred_linear))

# Visualisasi regresi linear sederhana
plt.scatter(X_test_simple, y_test, color='blue')
plt.plot(X_test_simple, y_pred_linear, color='red')
plt.title('Regresi Linear Sederhana (R&D Spend vs Profit)')
plt.xlabel('R&D Spend')
plt.ylabel('Profit')
plt.show()

# 2. Multiple Linear Regression
# Membuat dan melatih model regresi linear ganda
multiple_regressor = LinearRegression()
multiple_regressor.fit(X_train_multiple, y_train)

# Prediksi menggunakan model regresi linear ganda
y_pred_multiple = multiple_regressor.predict(X_test_multiple)

# Menampilkan hasil regresi linear ganda
print("Multiple Linear Regression:")
print("Mean Squared Error (MSE):", mean_squared_error(y_test, y_pred_multiple))
print("R-squared:", r2_score(y_test, y_pred_multiple))

# Visualisasi hasil prediksi multiple linear regression
plt.scatter(y_test, y_pred_multiple, color='blue')
plt.plot(y_test, y_pred_multiple, color='red') # Garis identitas (nilai prediksi sama dengan nilai aktual)
plt.title('Regresi Linear Ganda (Profit: Actual vs Predicted)')
plt.xlabel('Nilai Profit Sebenarnya')
plt.ylabel('Nilai Profit Prediksi')
plt.show()

# 3. Polynomial Regression
# Membuat fitur polinomial
poly_features = PolynomialFeatures(degree=2) # Degree polinomial dapat diubah
X_poly_train = poly_features.fit_transform(X_train_multiple)
X_poly_test = poly_features.transform(X_test_multiple)
```

```

poly_regressor = LinearRegression()
poly_regressor.fit(X_poly_train, y_train)

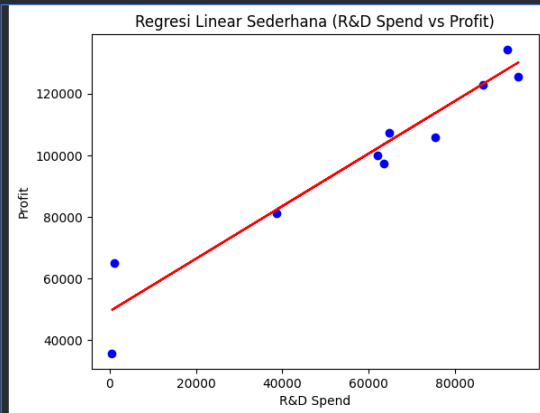
# Prediksi menggunakan model regresi polinomial
y_pred_poly = poly_regressor.predict(X_poly_test)

# Menampilkan hasil regresi polinomial
print("Polynomial Regression:")
print("Mean Squared Error (MSE):", mean_squared_error(y_test, y_pred_poly))
print("R-squared:", r2_score(y_test, y_pred_poly))

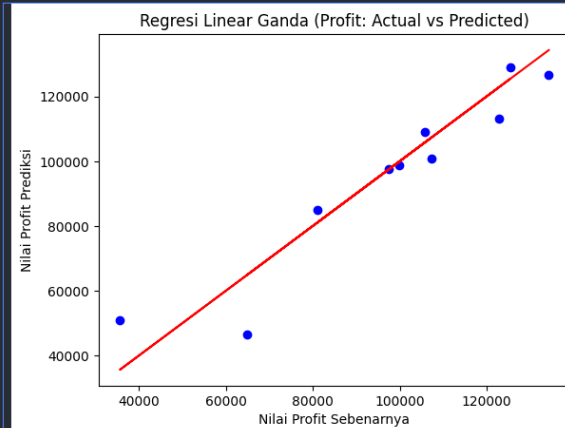
# Visualisasi regresi polinomial
plt.scatter(X_test_simple, y_test, color='blue')
plt.plot(X_test_simple, y_pred_poly, color='green')
plt.title('Regresi Polinomial (R&D Spend vs Profit, Degree = 2)')
plt.xlabel('R&D Spend')
plt.ylabel('Profit')
plt.show()

```

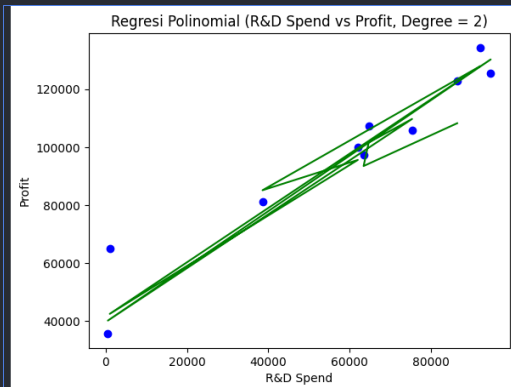
Simple Linear Regression:
Mean Squared Error (MSE): 59510962.80787997
R-squared: 0.9265108109341951



Multiple Linear Regression:
Mean Squared Error (MSE): 80926321.22295158
R-squared: 0.9000653083037321



Polynomial Regression:
Mean Squared Error (MSE): 90304882.29010044
R-squared: 0.8884838649039012



```
# Import library yang dibutuhkan
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Load dataset
data = pd.read_csv("/content/Position_Salaries.csv")

# Menampilkan 5 baris pertama dari dataset
data.head()

# Menisahkan fitur dan target
X = data[['Level']] # Fitur (Level)
y = data['Salary'] # Target (Salary)

# Membagi data menjadi set pelatihan dan pengujian (test_size bisa disesuaikan)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 1. Simple Linear Regression
# Membuat dan melatih model regresi linear sederhana
linear_regressor = LinearRegression()
linear_regressor.fit(X_train, y_train)

# Prediksi menggunakan model regresi linear sederhana
y_pred_linear = linear_regressor.predict(X_test)

# Menampilkan hasil regresi linear sederhana
print("Simple Linear Regression:")
print("Mean Squared Error (MSE):", mean_squared_error(y_test, y_pred_linear))
print("R-squared:", r2_score(y_test, y_pred_linear))

# Visualisasi regresi linear sederhana
plt.scatter(X, y, color='blue')
plt.plot(X, linear_regressor.predict(X), color='red')
plt.title('Regresi Linear Sederhana (Level vs Salary)')
plt.xlabel('Level')
plt.ylabel('Salary')
plt.show()

# 2. Multiple Linear Regression
# Untuk multiple regression, kita perlu lebih dari satu fitur. Misalkan menambah kolom dummy:
data = pd.get_dummies(data, drop_first=True)

# Menisahkan fitur dan target untuk multiple regression
X_multiple = data.drop(['Salary'], axis=1) # Menggunakan semua kolom selain Salary sebagai fitur
y_multiple = data['Salary']

# Membagi data untuk multiple regression
X_train_multiple, X_test_multiple, y_train_multiple, y_test_multiple = train_test_split(X_multiple, y_multiple, test_size=0.2, random_state=42)

# Membuat dan melatih model regresi linear ganda
multiple_regressor = LinearRegression()
multiple_regressor.fit(X_train_multiple, y_train_multiple)

# Prediksi menggunakan model regresi linear ganda
y_pred_multiple = multiple_regressor.predict(X_test_multiple)

# Menampilkan hasil regresi linear ganda
print("Multiple Linear Regression:")
print("Mean Squared Error (MSE):", mean_squared_error(y_test_multiple, y_pred_multiple))
print("R-squared:", r2_score(y_test_multiple, y_pred_multiple))
```

```

# Visualisasi hasil prediksi multiple linear regression
plt.scatter(y_test_multiple, y_pred_multiple, color='blue')
plt.plot(y_test_multiple, y_test_multiple, color='red') # Garis identitas (prediksi)
plt.title('Regresi Linear Ganda (Salary: Actual vs Predicted)')
plt.xlabel('Nilai Salary Sebenarnya')
plt.ylabel('Nilai Salary Prediksi')
plt.show()

# 3. Polynomial Regression
# Membuat fitur polinomial
poly_features = PolynomialFeatures(degree=4) # Degree polinomial dapat diubah
X_poly_train = poly_features.fit_transform(X_train)
X_poly_test = poly_features.transform(X_test)

# Membuat dan melatih model regresi polinomial
poly_regressor = LinearRegression()
poly_regressor.fit(X_poly_train, y_train)

# Prediksi menggunakan model regresi polinomial
y_pred_poly = poly_regressor.predict(X_poly_test)

# Menampilkan hasil regresi polinomial
print("Polynomial Regression:")
print("Mean Squared Error (MSE):", mean_squared_error(y_test, y_pred_poly))
print("R-squared:", r2_score(y_test, y_pred_poly))

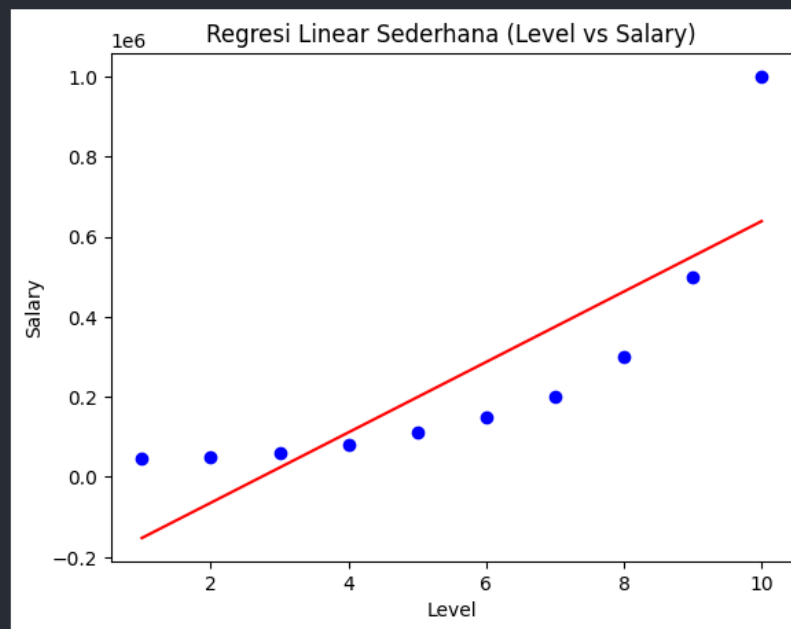
# Visualisasi regresi polinomial
plt.scatter(X, y, color='blue')
plt.plot(X, poly_regressor.predict(poly_features.transform(X)), color='green')
plt.title('Regresi Polinomial (Level vs Salary, Degree = 4)')
plt.xlabel('Level')
plt.ylabel('Salary')
plt.show()

```

Simple Linear Regression:

Mean Squared Error (MSE): 7840057409.334131

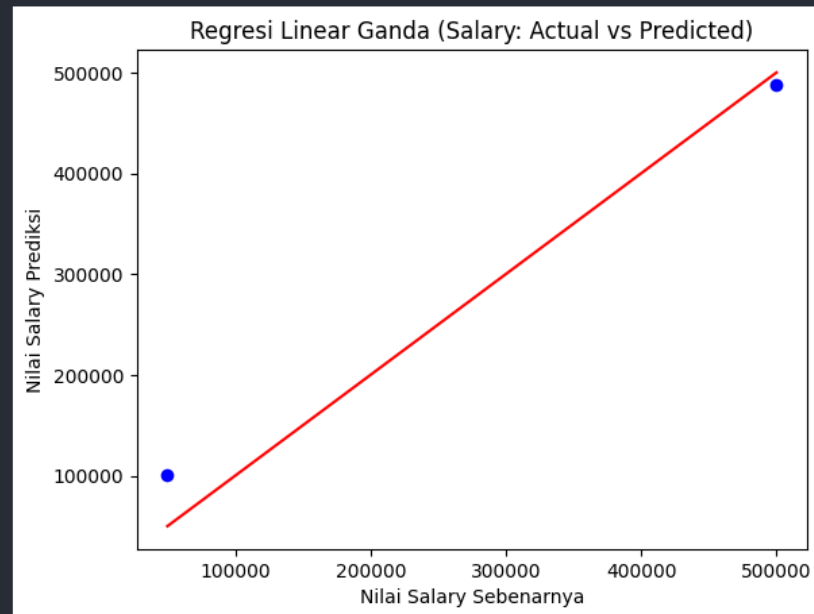
R-squared: 0.8451346684575974



Multiple Linear Regression:

Mean Squared Error (MSE): 1342487356.933715

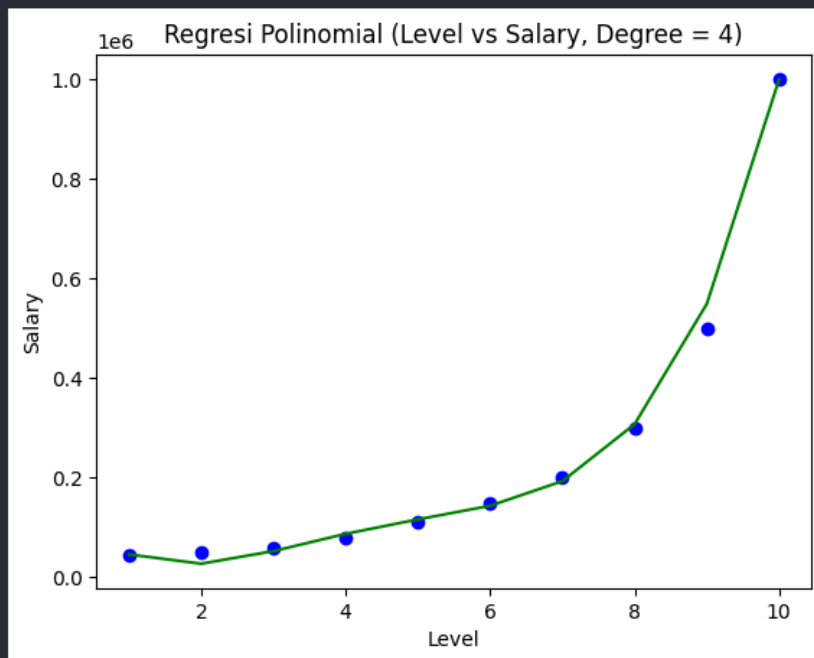
R-squared: 0.9734817312210624



Polynomial Regression:

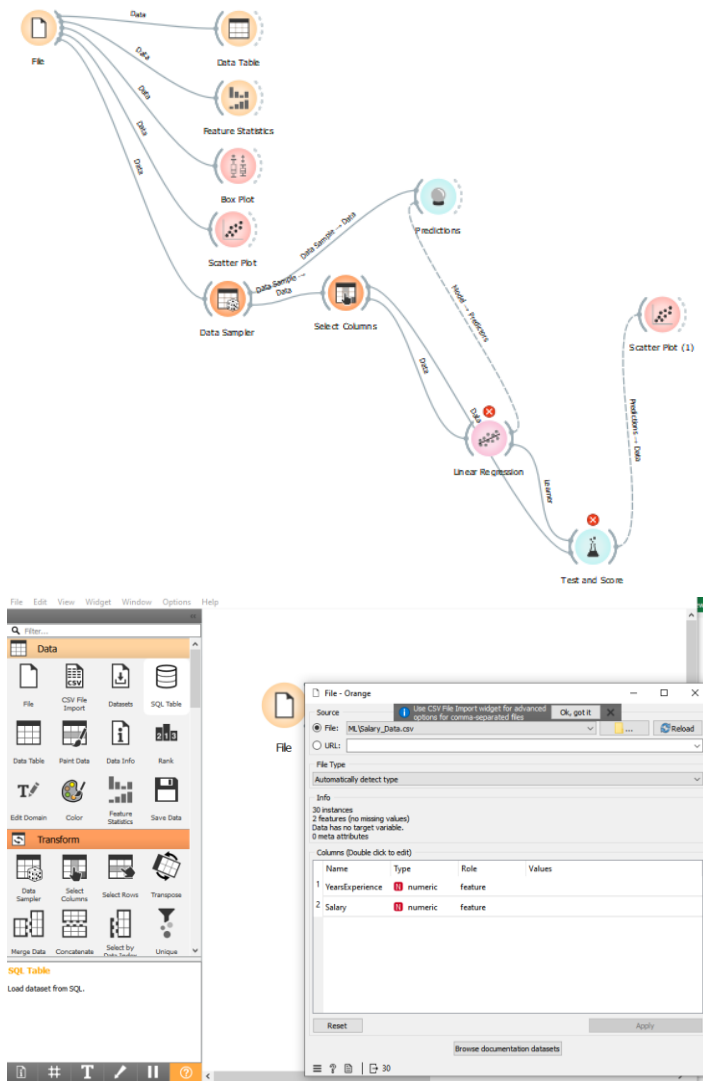
Mean Squared Error (MSE): 1444499305.547627

R-squared: 0.9714666803842444



Membuat regresi model dengan orange data mining

1. Simple Linear Regression dengan Sallary Dataset



Data Table

Data Table - Orange

Info

- 30 instances (no missing data)
- 2 features
- No target variable
- No meta attributes

Variables

- ☒ Show variable labels (if present)
- ☒ Visualize numeric values
- ☒ Color by instance classes

Selection

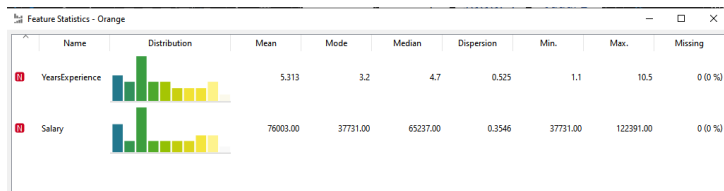
- ☒ Select full rows

Restore Original Order

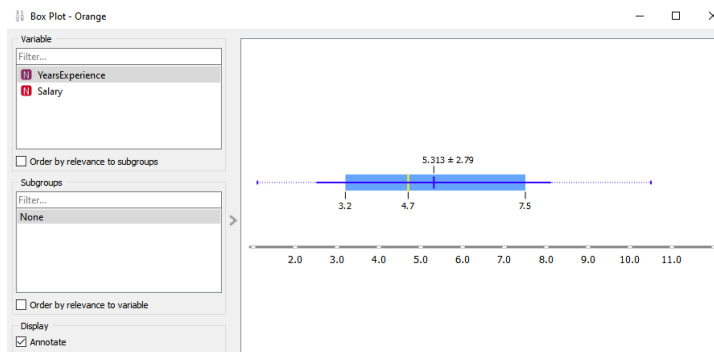
Send Automatically

YearsExperience	Salary
1.1	29943.00
1.3	46205.00
1.5	37731.00
2.0	45525.00
2.2	39891.00
2.9	56642.00
3.0	60150.00
3.2	54445.00
3.2	64445.00
3.7	57189.00
3.9	63218.00
4.0	55794.00
4.0	56957.00
4.1	57081.00
4.5	61111.00
4.9	67938.00
5.1	66029.00
5.3	83088.00
5.9	81363.00
6.0	93940.00
6.8	91738.00

Feature Statistics



Box Plot



Data Sampler

The Data Sampler window is configured with the following options:

- Sampling Type: ☒ Fixed proportion of data: 70 %
- ☐ Fixed sample size: Instances: 1
- ☐ Cross validation: Number of subsets: 10, Unused subset: 1
- ☐ Bootstrap
- Options: ☒ Replicable (deterministic) sampling, ☐ Stratify sample (when possible)

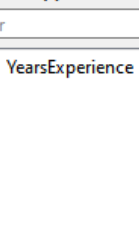
Buttons: Sample Data, 30, 21 | 9

Predictions

Linear Regression	YearsExperience	Salary
116171.39	9.6	112635.00
71665.83	4.9	67938.00
102914.41	8.2	113812.00
75453.53	5.3	83088.00
55568.07	3.2	64445.00
60302.70	3.7	57189.00
122799.88	10.3	122391.00
107649.05	8.7	109431.00
63143.48	4.0	56957.00
35682.60	1.1	39343.00

Buttons: 21 | 21 | -

Select Columns



Features (1)

Filter

N YearsExperience

Target (1)

N Salary

Metas

Linear Regression

Linear Regression - Orange

Name
Linear Regression

Parameters

☒ Fit intercept (unchecking it fixes it to zero)

Regularization

☒ No regularization

Regularization strength:
Alpha: 0.0001

☐ Ridge regression (L2)

☐ Lasso regression (L1)

Elastic net mixing:
L1 L2
0.50 : 0.50

☒ Apply Automatically

Test and Score

Test and Score - Orange

☒ Cross validation

Number of folds: 5

☒ Stratified

☐ Cross validation by feature

☐ Random sampling

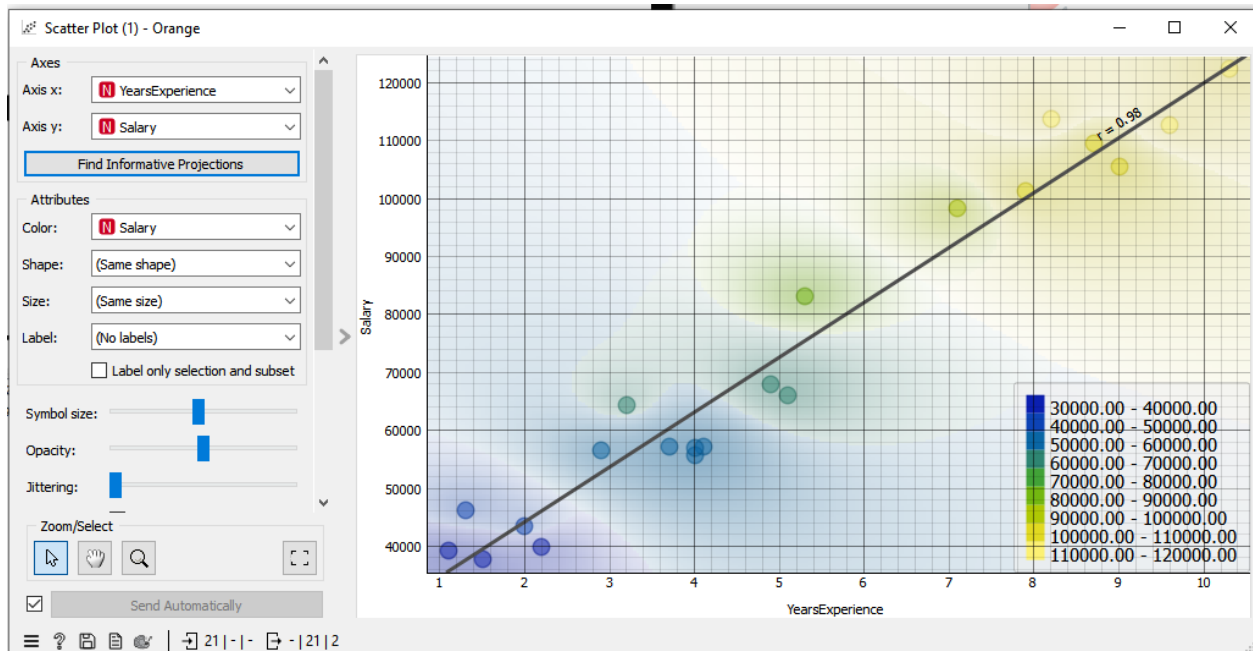
Repeat train/test: 10

Training set size: 66 %

☒ Stratified

Model	MSE	RMSE	MAE	MAPE	R2
Linear Regression	415...	6445...	568...	0.091	0.946

Scatter Plot



2. Multiple Linear Regression dengan 50 Start Up.csv

File - Orange

Source

Use CSV File Import widget for advanced options for comma-separated files

File: ML50_Startups.csv

File Type: Automatically detect type

Info

50 instances
5 features (no missing values)
Data has no target variable.
0 meta attributes

Columns (Double click to edit)

Name	Type	Role	Values
1 R&D Spend	numeric	feature	
2 Administration	numeric	feature	
3 Marketing Spend	numeric	feature	
4 State	categorical	feature	California, Florida, New York
5 Profit	numeric	feature	

Reset Apply

Browse documentation datasets

50

Data Table - Orange

Info
50 instances (no missing data)
5 features
No target variable.
No meta attributes.

Variables

- ☒ Show variable labels (if present)
- ☒ Visualize numeric values
- ☒ Color by instance classes

Selection

- ☒ Select full rows

Restore Original Order

☒ Send Automatically

	R&D Spend	Administration	Marketing Spend	State	Profit
1	165349.20	136897.80	471784.10	New York	192261.83
2	162597.70	151377.59	443898.53	California	191792.06
3	153441.51	101145.55	407934.54	Florida	191050.39
4	144372.41	118671.85	383199.62	New York	182901.99
5	142107.34	91391.77	366168.42	Florida	166187.94
6	131876.90	99814.71	362861.36	New York	156991.12
7	134615.46	147198.87	127716.82	California	156122.51
8	130298.13	145530.06	323876.68	Florida	155752.60
9	120542.52	148718.95	311613.29	New York	152211.77
10	123334.88	108679.17	304981.62	California	149759.96
11	101913.08	110594.11	229160.95	Florida	146121.95
12	100671.96	91790.61	249744.55	California	144259.40
13	93863.75	127320.38	249839.44	Florida	141585.52
14	91992.39	135495.07	252664.93	California	134307.35
15	119943.24	156547.42	256512.92	Florida	132602.65
16	114523.61	122616.84	261776.23	New York	129917.04
17	78013.11	121597.55	264346.06	California	126992.93
18	94657.16	145077.58	282574.31	New York	125370.37
19	91749.16	114175.79	294919.57	Florida	124266.90
20	86419.70	153514.11	0.00	New York	122776.86
21	76253.86	113867.30	298664.47	California	118474.03
22	78389.47	153773.43	299737.29	New York	111313.02
23	73994.56	122782.75	303319.26	Florida	110352.25
24	67633.63	105761.03	304766.73	California	100733.00

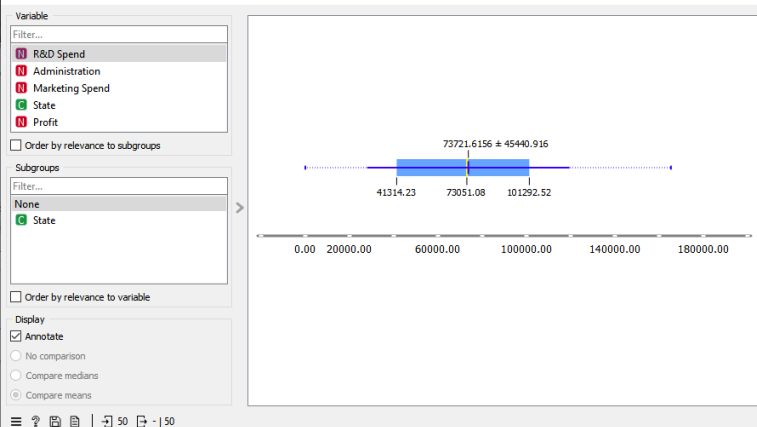
Feature Statistics - Orange

Name	Distribution	Mean	Mode	Median	Dispersion	Min.	Max.	Missing
R&D Spend		73721.6156	0.00	73051.08	0.6164	0.00	165349.20	0 (0 %)
Administration		121344.6396	51283.14	122699.7950	0.2286	51283.14	182645.56	0 (0 %)
Marketing Spend		211025.0978	0.00	212716.24	0.5737	0.00	471784.10	0 (0 %)
Profit		112012.6392	14681.40	107978.19	0.3562	14681.40	192261.83	0 (0 %)
State			California		1.1			0 (0 %)

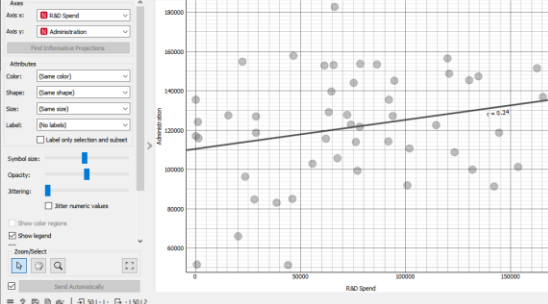
Color: None

☒ Send Automatically

Box Plot - Orange



Scatter Plot - Orange



Data Sampler - Orange

Sampling Type

☒ Fixed proportion of data:

70 %

☐ Fixed sample size

Instances: 1

☐ Sample with replacement

☐ Cross validation

Number of subsets: 10

Unused subset: 1

☐ Bootstrap

Options

☒ Replicable (deterministic) sampling

☐ Stratify sample (when possible)

Sample Data

Select Columns - Orange

Ignored

Filter

Features (4)

Filter

R&D Spend

Administration

Marketing Spend

State

Target (1)

Profit

Metas

Linear Regression - Orange

Name

Linear Regression

Parameters

☒ Fit intercept (unchecking it fixes it to zero)

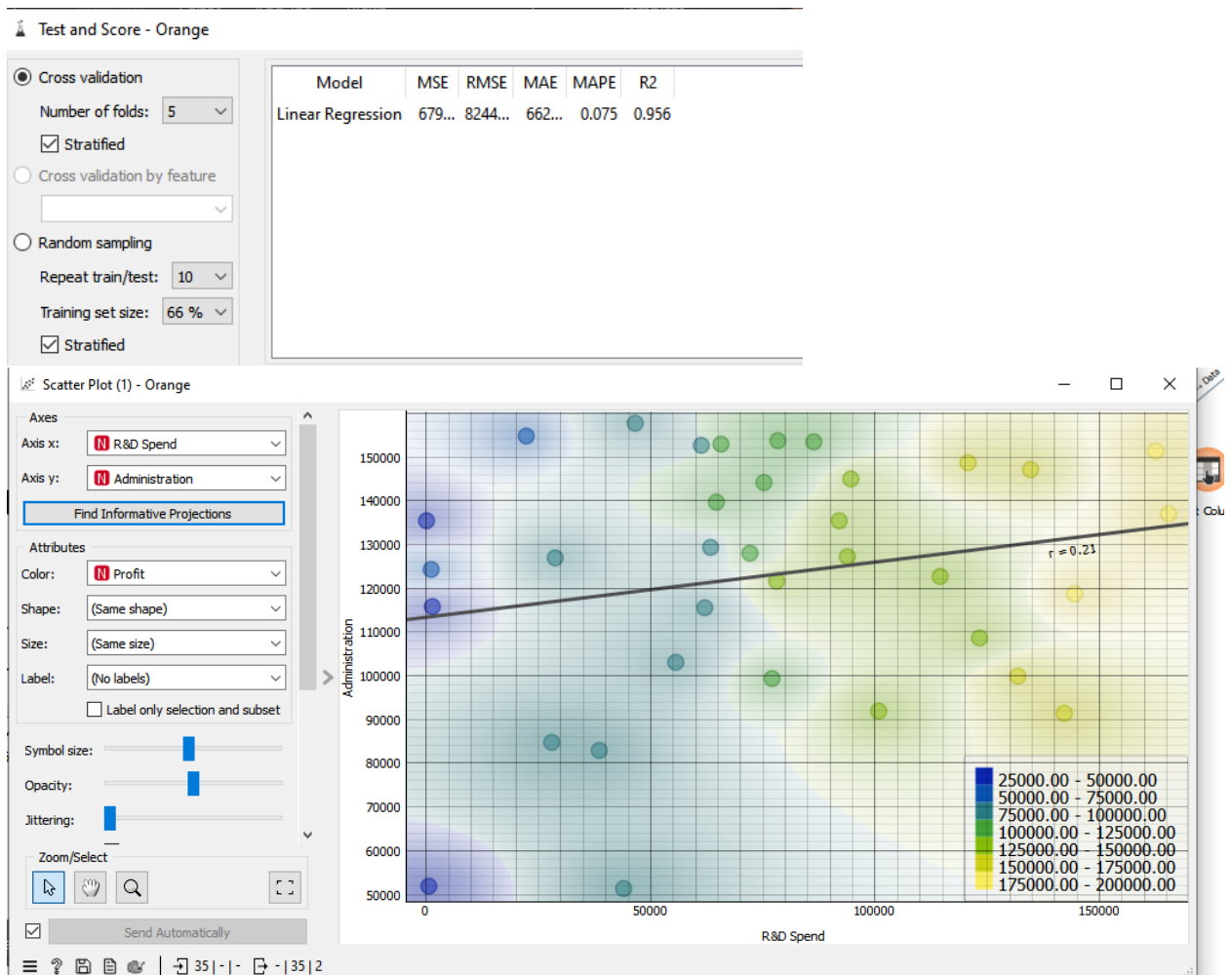
Regularization

☒ No regularization Regularization strength:

Predictions - Orange

Restore Original Order

	Linear Regression	R&D Spend	Administration	Marketing Spend	State
1	130096.33	91992.39	135495.07	252664.93	California
2	83213.05	38558.51	82982.09	174999.30	California
3	101454.69	61994.48	115641.28	91131.24	Florida
4	46485.03	1000.23	124153.04	1903.93	New York
5	129168.83	94657.16	145077.58	282574.31	New York
6	42915.66	542.05	51743.15	0.00	New York
7	114412.77	75328.87	144135.98	134050.07	Florida
8	105594.44	64664.71	139553.16	137962.62	California
9	101847.89	63408.86	129219.61	46085.25	California
10	115906.52	86419.70	153514.11	0.00	New York



3. Regression menggunakan Position Salaries.csv

File - Orange

Source

☒ File: ML\Position_Salaries.csv

☐ URL:

File Type

Automatically detect type

Info

10 instances

2 features (no missing values)

Data has no target variable.

1 meta attribute

Columns (Double click to edit)

	Name	Type	Role	Values
1	Level	numeric	feature	
2	Salary	numeric	feature	
3	Position	text	meta	

Reset Apply



Browse documentation datasets

Data Table - Orange

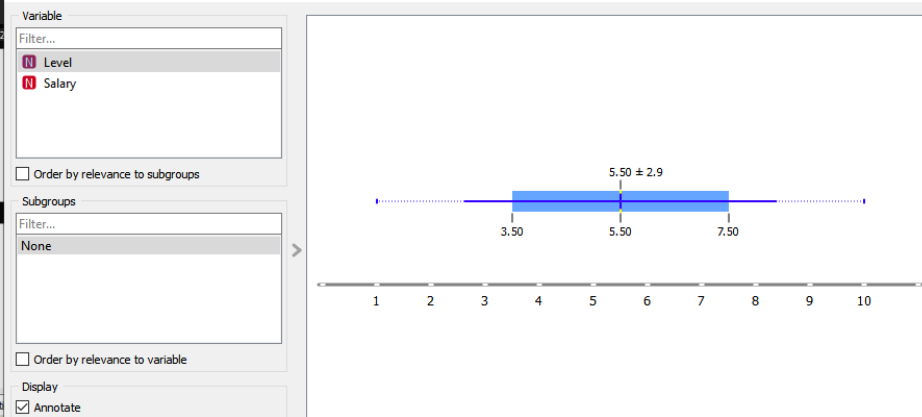
Info			
10 instances (no missing data)			
2 features			
No target variable.			
1 meta attribute			
Variables			
<input checked="" type="checkbox"/> Show variable labels (if present)			
<input checked="" type="checkbox"/> Visualize numeric values			
<input checked="" type="checkbox"/> Color by instance classes			
Selection			
<input checked="" type="checkbox"/> Select full rows			

	Position	Level	Salary
1	Business Analyst	1	45000
2	Junior Consultant	2	50000
3	Senior Consulta...	3	60000
4	Manager	4	80000
5	Country Manager	5	110000
6	Region Manager	6	150000
7	Partner	7	200000
8	Senior Partner	8	300000
9	C-level	9	500000
10	CEO	10	1000000

Feature Statistics - Orange

	Name	Distribution	Mean	Mode	Median	Dispersion	Min.	Max.	Missing
N	Level		5.50	1	5.50	0.52	1	10	0 (0 %)
N	Salary		249500	45000	130000	1.14	45000	1000000	0 (0 %)

Box Plot - Orange



Data Sampler - Orange

Sampling Type

☒ Fixed proportion of data:

70 %

☐ Fixed sample size

Instances:

☐ Sample with replacement

☐ Cross validation

Number of subsets:

Unused subset:

☐ Bootstrap

Options

☒ Replicable (deterministic) sampling

☐ Stratify sample (when possible)

Sample Data

10 7/3

Select Columns - Orange

Ignored

Filter

Features (1)

Filter

Level

Target (1)

Salary

Metas (1)

Position

Linear Regression - Orange

Name

Linear Regression

Parameters

☒ Fit intercept (unchecking it fixes it to zero)

Regularization

☒ No regularization
 Regularization strength:

☐ Ridge regression (L2)
 Alpha: 0.0001

☐ Lasso regression (L1)
 Elastic net mixing:

L1

L2

☐ Elastic net regression
 0.50 : 0.50

☒ Apply Automatically

Predictions - Orange

Restore O

	Linear Regression	Position	Level	Salary
1	582941	C-level	9	500000
2	6728	Junior Consultant	2	50000
3	335993	Region Manager	6	150000
4	-75588	Business Analyst	1	45000
5	500625	Senior Partner	8	300000
6	89044	Senior Consulta...	3	60000
7	665257	CEO	10	1000000

Test and Score - Orange

Cross validation

Number of folds: 5

☒ Stratified

☐ Cross validation by feature

☐ Random sampling

Repeat train/test: 10

Training set size: 66 %

☒ Stratified

Model	MSE	RMSE	MAE	MAPE	R2
Linear Regression	701...	2648...	214...	1.470	0.335

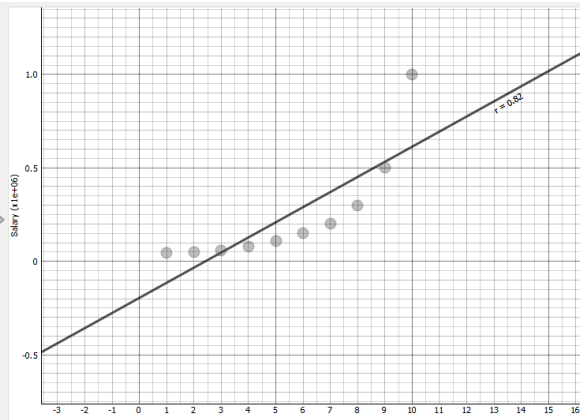
Scatter Plot - Orange

Axes
 Axis x: Level
 Axis y: Salary
 Find Informative Projections

Attributes
 Color: (Same color)
 Shape: (Same shape)
 Size: (Same size)
 Label: (No labels)
☐ Label only selection and subset

Symbol size:
 Opacity:
 Jittering:
☐ Jitter numeric values

☐ Show color regions
☒ Show legend
 Zoom/Select



Scatter Plot (1) - Orange

Axes
 Axis x: Level
 Axis y: Salary
 Find Informative Projections

Attributes
 Color: Salary
 Shape: Fold
 Size: (Same size)
 Label: Position
☐ Label only selection and subset

Symbol size:
 Opacity:
 Jittering:
☒ Send Automatically

