IMPORTING LIBRARIES

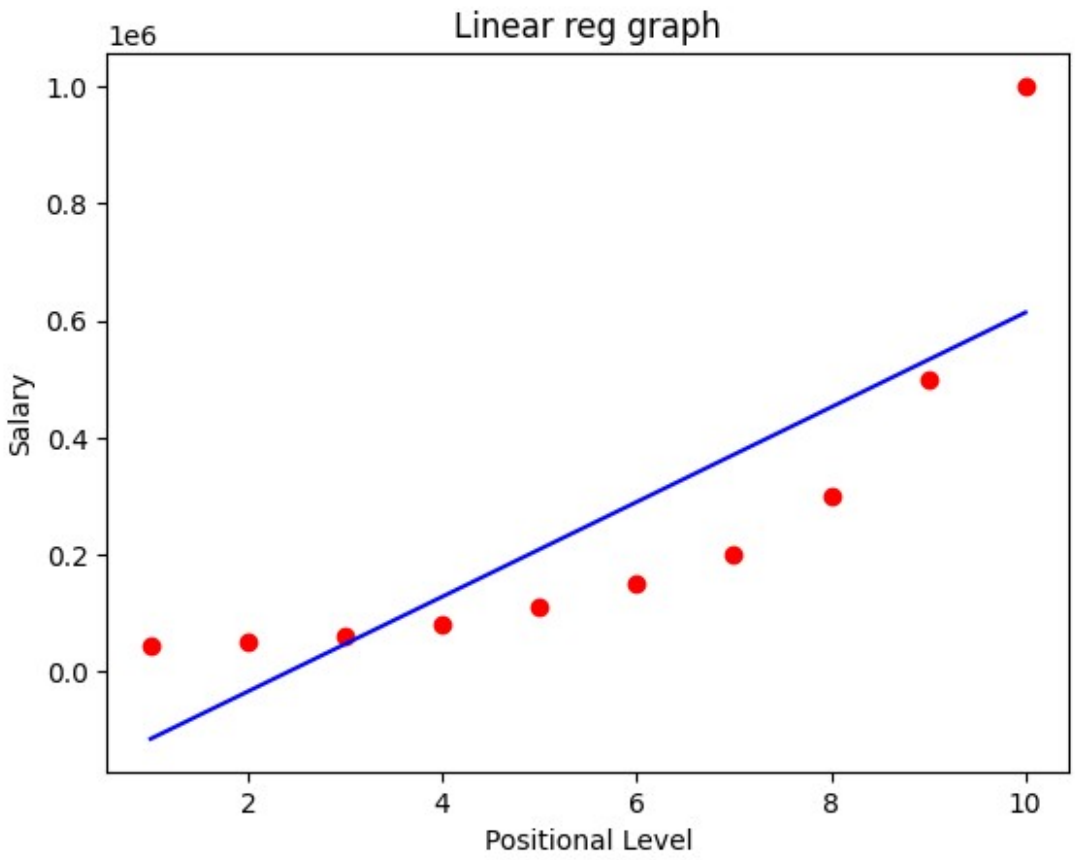
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv(r"C:\Users\mohap\Downloads\Polynomial_regression\
Polynomial regression\Polynomial regression\1.POLYNOMIAL REGRESSION\
emp_sal.csv")
data
```

	Position	Level	Salary
0	Jr Software Engineer	1	45000
1	Sr Software Engineer	2	50000
2	Team Lead	3	60000
3	Manager	4	80000
4	Sr manager	5	110000
5	Region Manager	6	150000
6	AVP	7	200000
7	VP	8	300000
8	CTO	9	500000
9	CEO	10	1000000

```
X = data.iloc[:,1:2].values
y = data.iloc[:,2].values

from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X,y)
```

```
LinearRegression()
```

```
plt.scatter(X,y, color = 'red')
plt.plot(X,lin_reg.predict(X),color = 'blue')
plt.title("Linear reg graph")
plt.xlabel("Positional Level")
plt.ylabel("Salary")
plt.show()
```



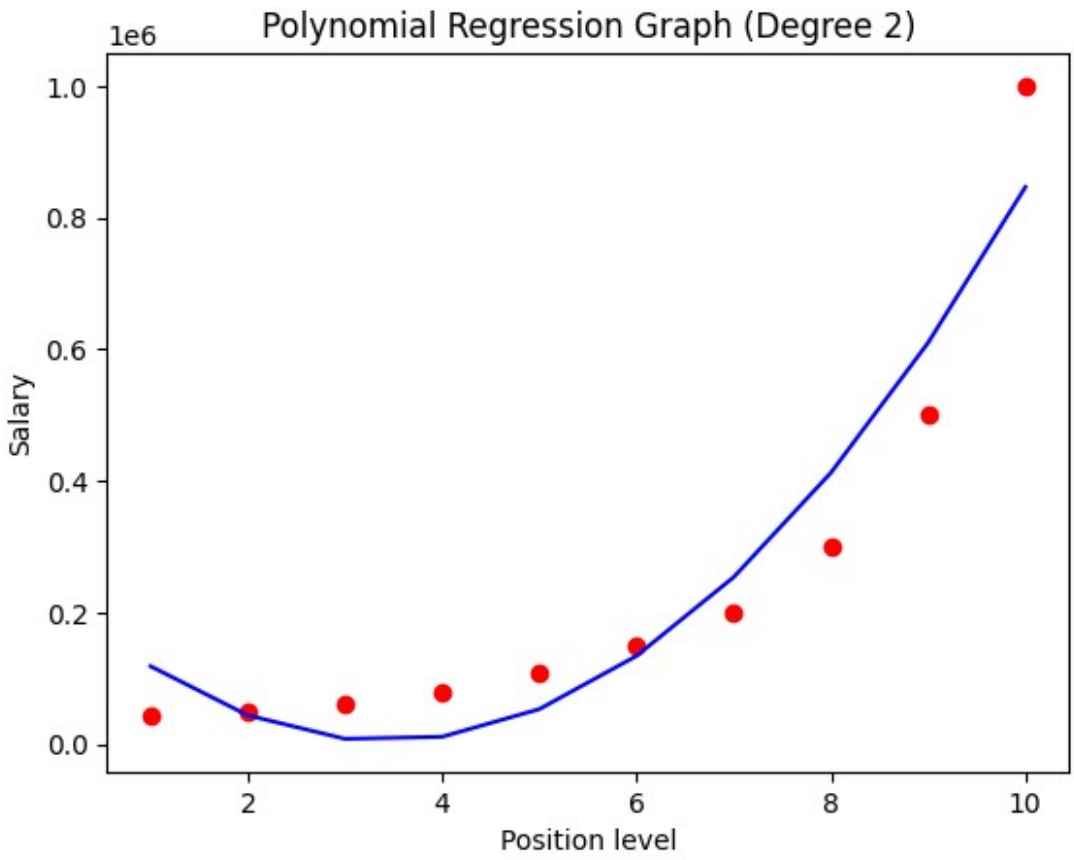

```
lin_model_pred = lin_reg.predict(([[6.5]]))
print(f"Linear Regression Prediction for 6.5: {lin_model_pred}")
```

```
Linear Regression Prediction for 6.5: [330378.78787879]
```

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=2)
X_poly = poly_reg.fit_transform(X)

# Create and train the Polynomial Regression model
poly_regressor = LinearRegression()
poly_regressor.fit(X_poly, y)
```

```
LinearRegression()
```

```
plt.scatter(X,y, color = 'red')
plt.plot(X,poly_regressor.predict(X_poly), color = 'blue') # Use
X_poly for prediction
plt.title("Polynomial Regression Graph (Degree 2)")
plt.xlabel("Position level")
plt.ylabel("Salary")
plt.show()
```



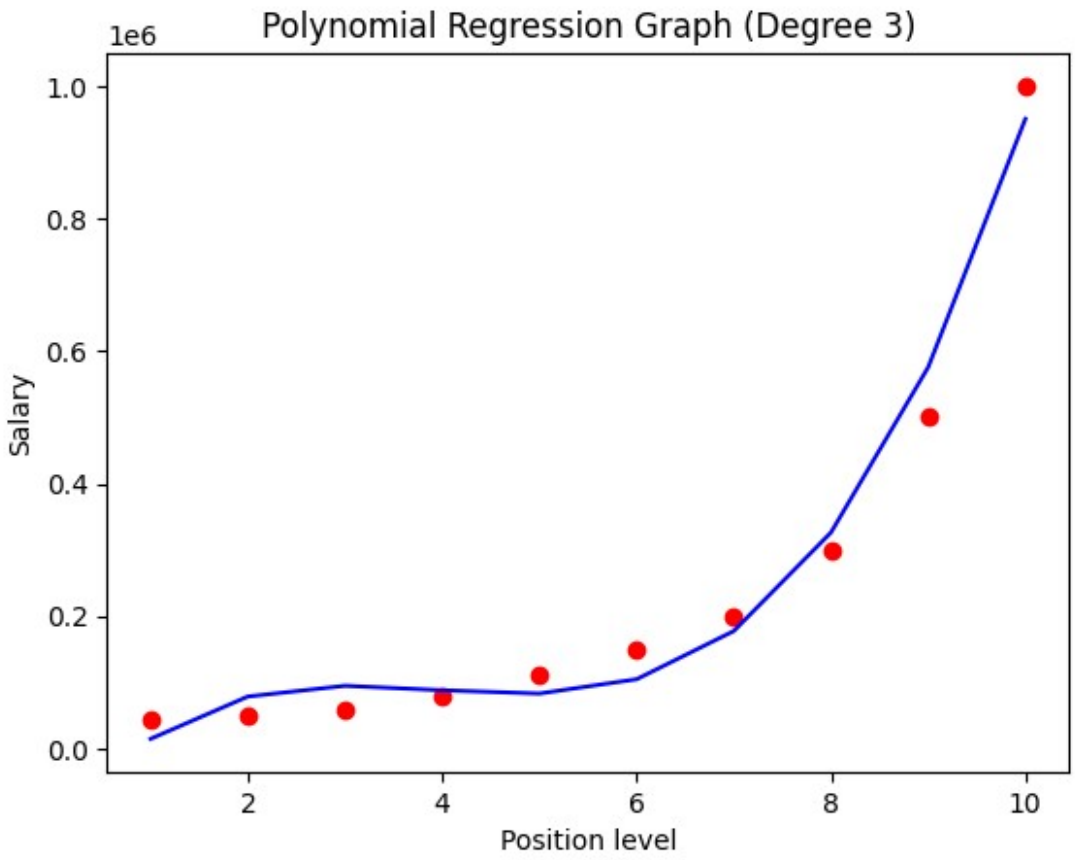

```
poly_model_pred = poly_regressor.predict(poly_reg.transform([[6.5]]))
# Use poly_reg.transform
print(f"Polynomial Regression Prediction for 6.5: {poly_model_pred}")
```

```
Polynomial Regression Prediction for 6.5: [189498.10606061]
```

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=3)
X_poly = poly_reg.fit_transform(X)

# Create and train the Polynomial Regression model
poly_regressor = LinearRegression()
poly_regressor.fit(X_poly, y)
```

```
LinearRegression()
```

```
plt.scatter(X,y, color = 'red')
plt.plot(X,poly_regressor.predict(X_poly), color = 'blue') # Use
X_poly for prediction
plt.title("Polynomial Regression Graph (Degree 3)")
plt.xlabel("Position level")
plt.ylabel("Salary")
plt.show()
```




```
poly_model_pred = poly_regressor.predict(poly_reg.transform([[6.5]]))
# Use poly_reg.transform
print(f"Polynomial Regression Prediction for 6.5: {poly_model_pred}")
```

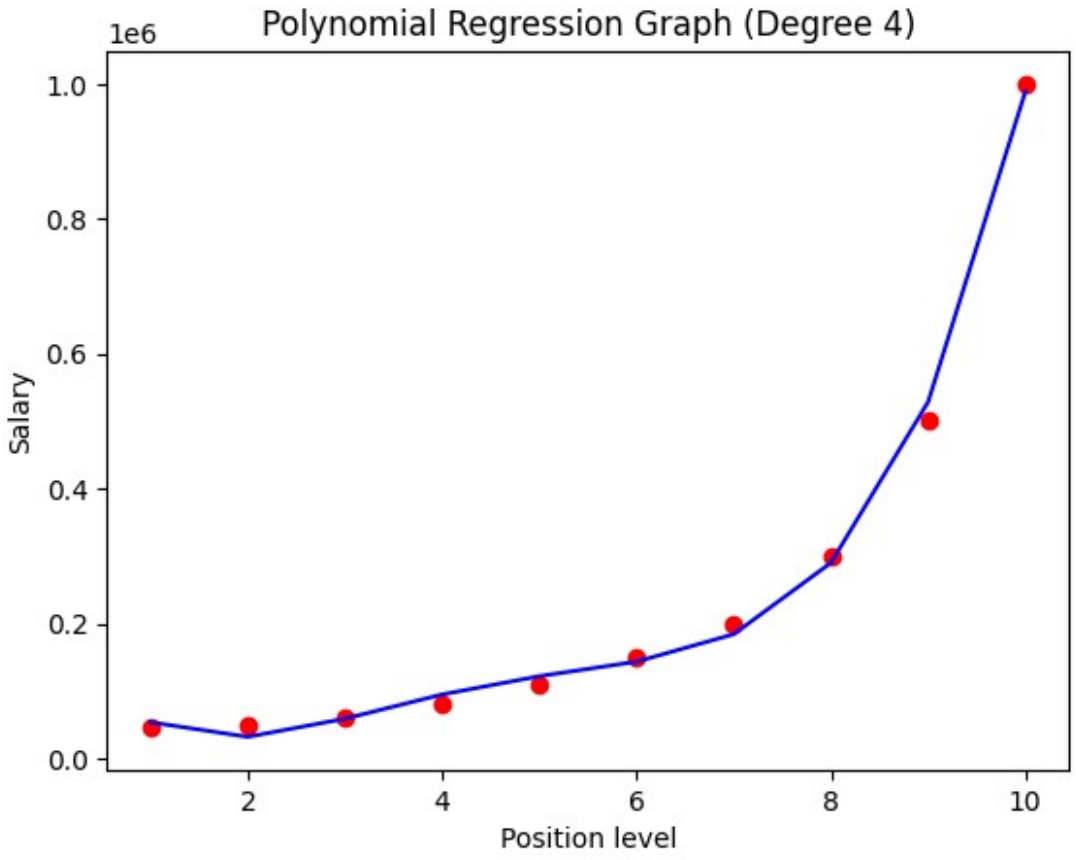
```
Polynomial Regression Prediction for 6.5: [133259.46969697]
```

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=4)
X_poly = poly_reg.fit_transform(X)

# Create and train the Polynomial Regression model
poly_regressor = LinearRegression()
poly_regressor.fit(X_poly, y)
```

```
LinearRegression()
```

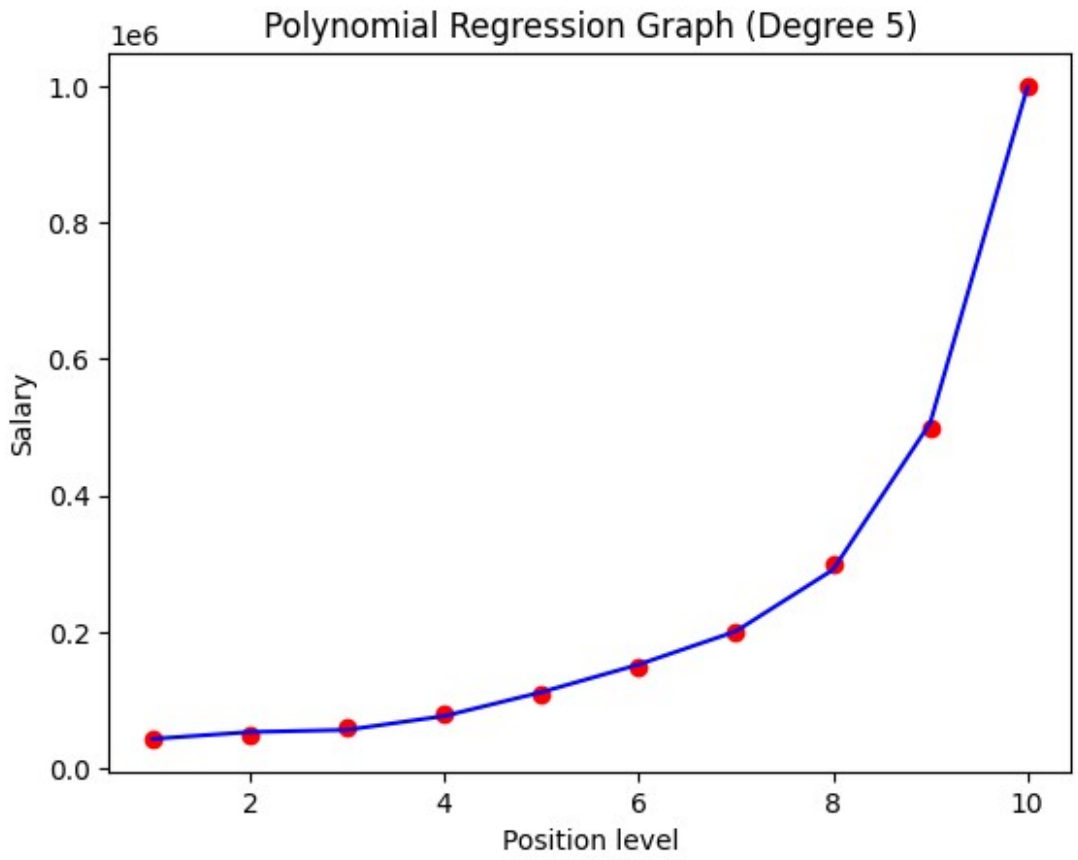
```
plt.scatter(X,y, color = 'red')
plt.plot(X,poly_regressor.predict(X_poly), color = 'blue') # Use
X_poly for prediction
plt.title("Polynomial Regression Graph (Degree 4)")
plt.xlabel("Position level")
plt.ylabel("Salary")
plt.show()
```

```
poly_model_pred = poly_regressor.predict(poly_reg.transform([[6.5]]))
# Use poly_reg.transform
print(f"Polynomial Regression Prediction for 6.5: {poly_model_pred}")
```




```
Polynomial Regression Prediction for 6.5: [158862.45265155]
```

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=5)
X_poly = poly_reg.fit_transform(X)

# Create and train the Polynomial Regression model
poly_regressor = LinearRegression()
poly_regressor.fit(X_poly, y)

plt.scatter(X,y, color = 'red')
plt.plot(X,poly_regressor.predict(X_poly), color = 'blue') # Use
X_poly for prediction
plt.title("Polynomial Regression Graph (Degree 5)")
plt.xlabel("Position level")
plt.ylabel("Salary")
plt.show()
```

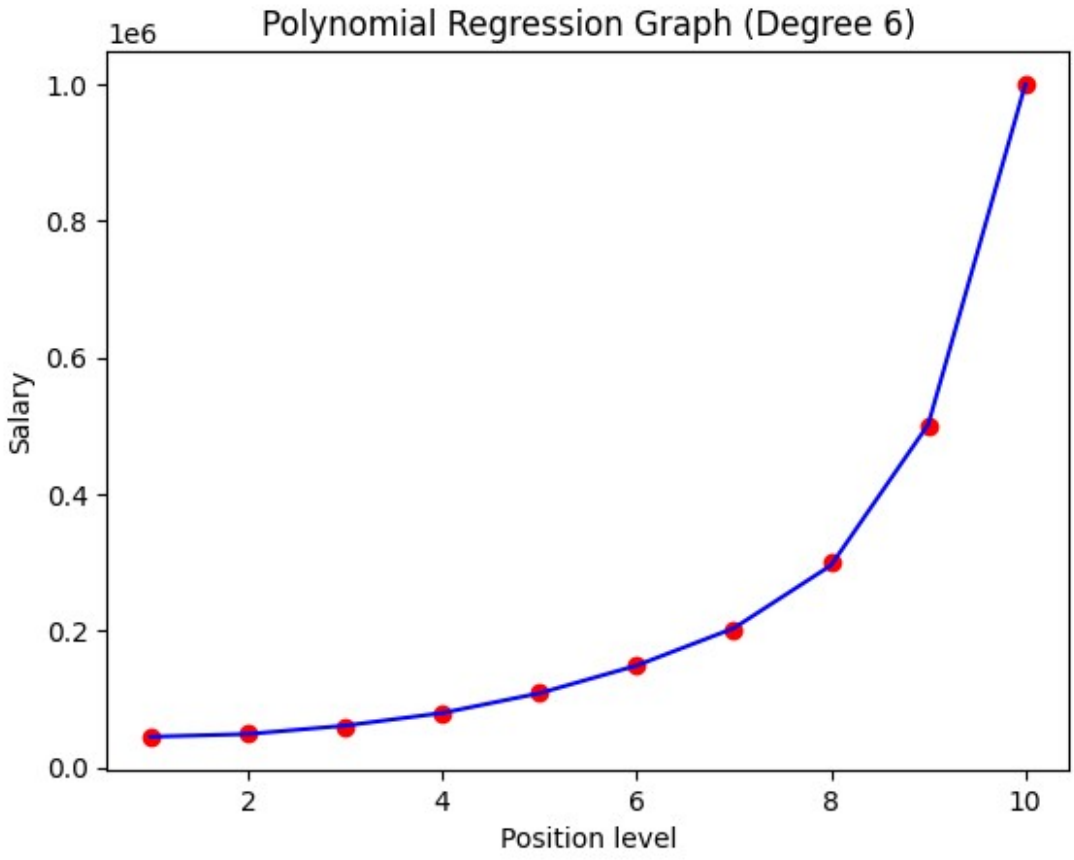



```
poly_model_pred = poly_regressor.predict(poly_reg.transform([[6.5]]))
# Use poly_reg.transform
print(f"Polynomial Regression Prediction for 6.5: {poly_model_pred}")
```

```
Polynomial Regression Prediction for 6.5: [174878.07765173]
```

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=6)
X_poly = poly_reg.fit_transform(X)

# Create and train the Polynomial Regression model
poly_regressor = LinearRegression()
poly_regressor.fit(X_poly, y)

plt.scatter(X,y, color = 'red')
plt.plot(X,poly_regressor.predict(X_poly), color = 'blue') # Use
X_poly for prediction
plt.title("Polynomial Regression Graph (Degree 6)")
plt.xlabel("Position level")
plt.ylabel("Salary")
plt.show()

poly_model_pred = poly_regressor.predict(poly_reg.transform([[6.5]]))
```

```
# Use poly_reg.transform
print(f"Polynomial Regression Prediction for 6.5: {poly_model_pred}")
```

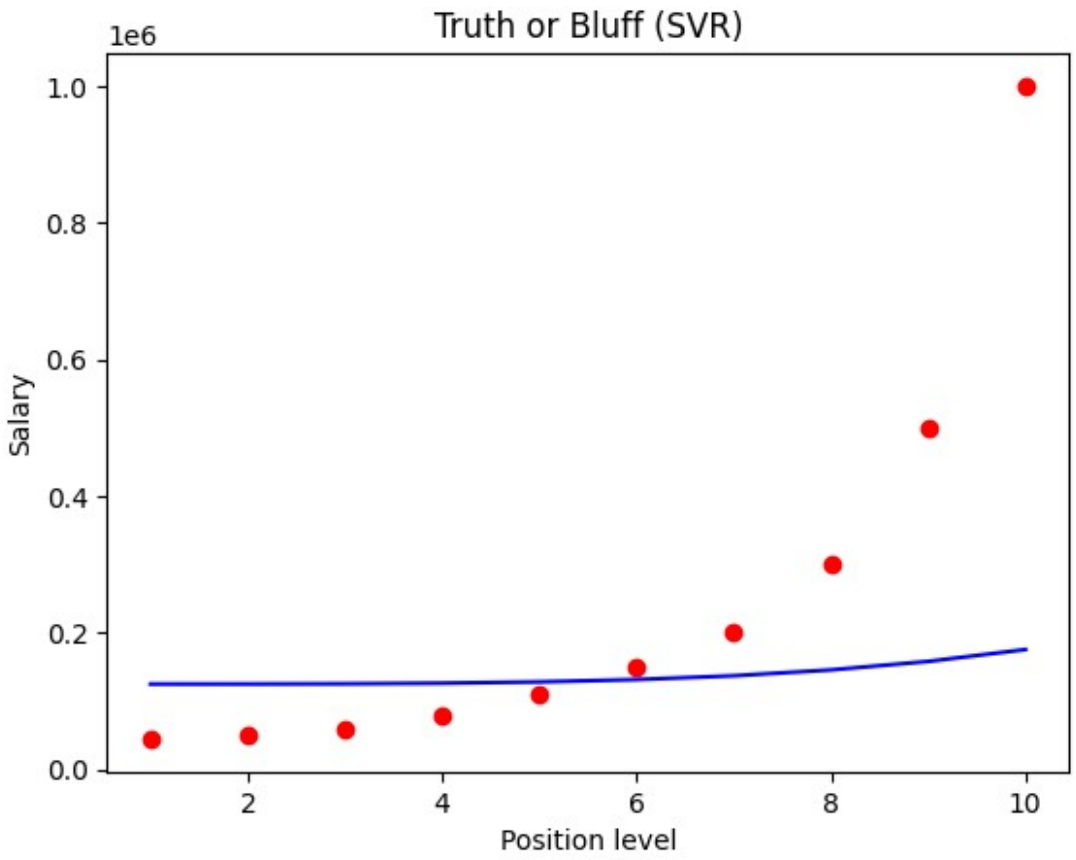



```
Polynomial Regression Prediction for 6.5: [174192.81930603]
```

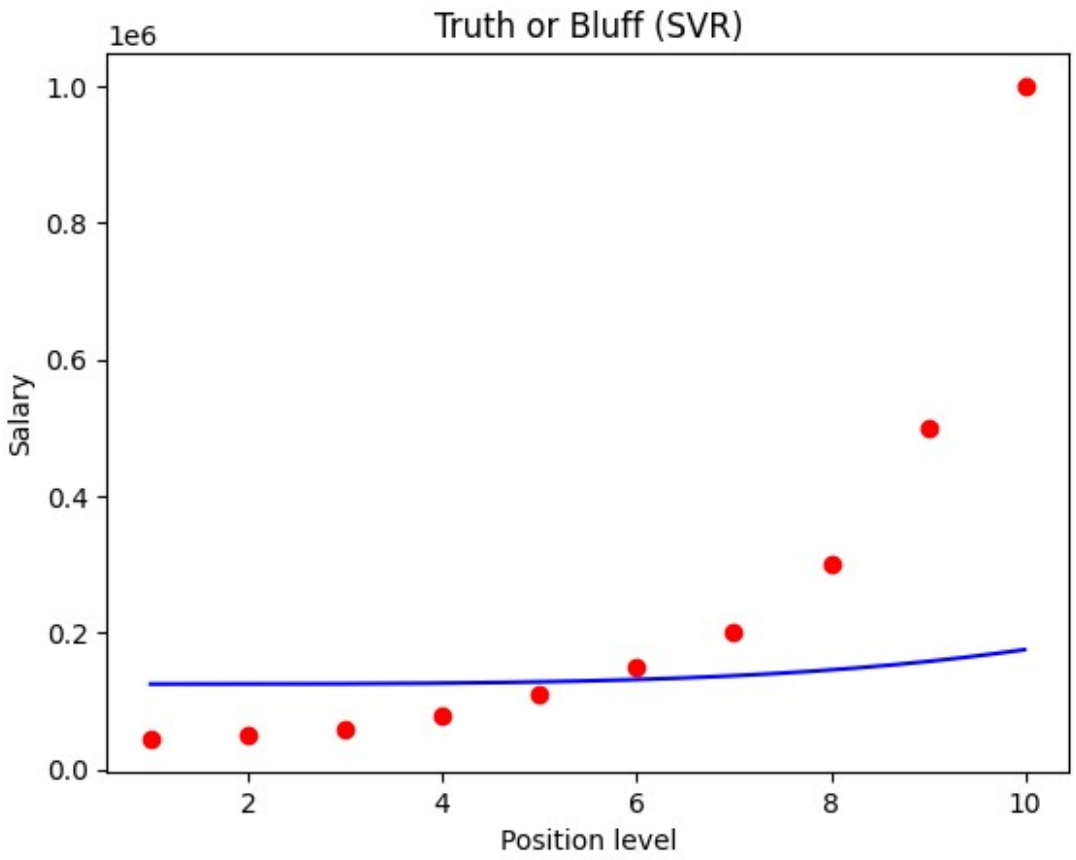
```
# svm model
from sklearn.svm import SVR
svr_regressor = SVR(kernel='poly',degree = 5,gamma = 'scale' )
svr_regressor.fit(X,y)

svr_model_pred = svr_regressor.predict([[6.5]])
print(svr_model_pred)


# Fitting SVR to the dataset
from sklearn.svm import SVR
regressor = SVR(kernel = 'poly',degree = 4)
regressor.fit(X, y)

y_pred_svr = regressor.predict([[6.5]])


# Visualising the SVR results
```

```
plt.scatter(X, y, color = 'red')
plt.plot(X, regressor.predict(X), color = 'blue')
plt.title('Truth or Bluff (SVR)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()


# Visualising the SVR results (for higher resolution and smoother
curve)
X_grid = np.arange(min(X), max(X), 0.01) # choice of 0.01 instead of
0.1 step because the data is feature scaled
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X, y, color = 'red')
plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
plt.title('Truth or Bluff (SVR)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

```
[164079.01344549]
```




```
C:\Users\mohap\AppData\Local\Temp\ipykernel_16604\3788301405.py:28:
DeprecationWarning: Conversion of an array with ndim > 0 to a scalar
is deprecated, and will error in future. Ensure you extract a single
element from your array before performing this operation. (Deprecated
NumPy 1.25.)
  X_grid = np.arange(min(X), max(X), 0.01) # choice of 0.01 instead of
0.1 step because the data is feature scaled
```




```
# knn model
from sklearn.neighbors import KNeighborsRegressor
knn_reg_model = KNeighborsRegressor(n_neighbors=5, weights='distance',
p=2)
knn_reg_model.fit(X,y)

knn_reg_pred = knn_reg_model.predict([[6.5]])
print(knn_reg_pred)
```

```
[175348.8372093]
```

```
# Plotting the actual data points
plt.scatter(X, y, color='red', label='Actual Data')

# Plotting Linear Regression predictions
plt.plot(X, lin_reg.predict(X), color='blue', label='Linear
Regression')

# Plotting Polynomial Regression predictions
plt.plot(X, poly_regressor.predict(X_poly), color='green',
label='Polynomial Regression (Degree 6)')

# Plotting SVR predictions
plt.plot(X, svr_regressor.predict(X), color='orange', label='SVR
```

```
(Degree 5)')

# Plotting KNN predictions
plt.plot(X, knn_reg_model.predict(X), color='purple', label='KNN
Regression')

# Adding labels and title
plt.title('Model Predictions Comparison')
plt.xlabel('Position Level')
plt.ylabel('Salary')
plt.legend()
plt.show()
```