Creado por:

Isabel Maniega

Ejercicio 2

In [1]:
```python
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [2]:
```python
# Importing the dataset
dataset = pd.read_csv('Position_Salaries.csv')
```

In [3]:
```python
dataset
```

Out[3]:

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

In [4]:
```python
X = dataset.iloc[:, 1:2].values
y = dataset.iloc[:, 2].values
```

In [5]:
```python
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

Lineal y Polinomial Regression

In [6]:
```python
# Fitting Linear Regression to the dataset
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X, y)
```

Out[6]:
```
▾ LinearRegression
LinearRegression()
```

In [7]:
```python
# Fitting Polynomial Regression to the dataset
from sklearn.preprocessing import PolynomialFeatures

poly_reg = PolynomialFeatures(degree = 4)
X_poly = poly_reg.fit_transform(X)
poly_reg.fit(X_poly, y)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(X_poly, y)
```

Out[7]:
```
▾ LinearRegression
LinearRegression()
```

Probaremos con un nivel de 6.5 para saber el sueldo que debería de ganar, viendo el sueldo en la tabla, debería estar entre 200.000 y 300.000, entre el puesto de Partner y Senior Partner

In [33]:
```python
# Predicting a new result with Linear Regression

lin_reg.predict([[6.5]])
```

Out[33]:
```
array([330378.78787879])
```

In [34]:
```python
# Predicting a new result with Polynomial Regression

lin_reg_2.predict(poly_reg.fit_transform([[6.5]]))
```

Out[34]:
```
array([158862.45265157])
```

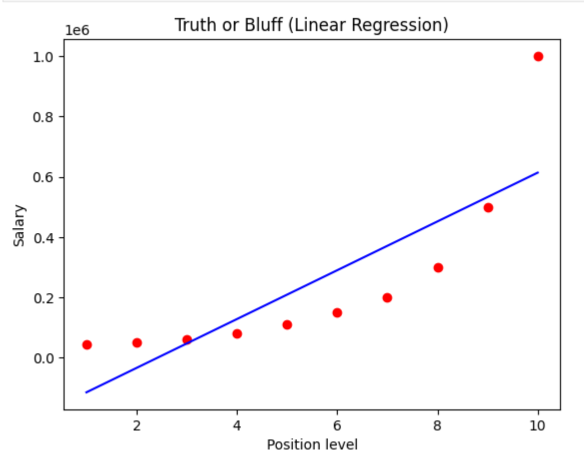
In [9]:
```python
print("Valor de pendiente o coeficiente 'a':")
print(lin_reg_2.coef_)
```

```
Valor de pendiente o coeficiente 'a':
[      0.         -211002.33100293   94765.44289063  -15463.28671331
    890.15151515]
```

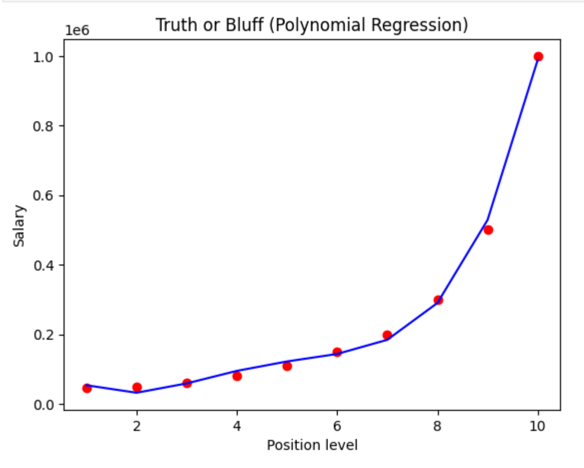
In [10]:
```python
print("Precisión del modelo: ")
print(lin_reg_2.score(X_poly, y))
```

```
Precisión del modelo:
0.9973922891706615
```

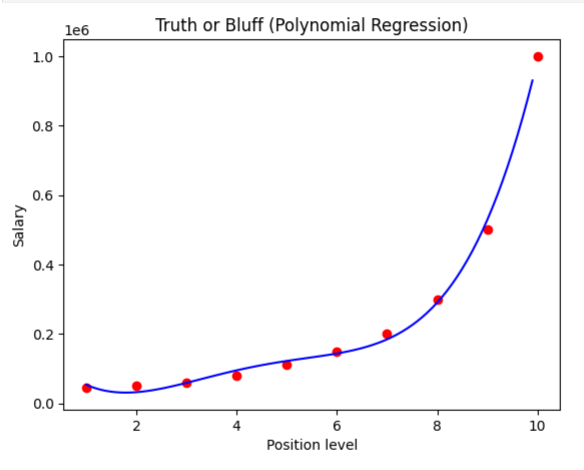
In [11]:
```python
# Visualising the Linear Regression results
plt.scatter(X, y, color = 'red')
plt.plot(X, lin_reg.predict(X), color = 'blue')
plt.title('Truth or Bluff (Linear Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```



In [12]:
```python
# Visualising the Polynomial Regression results
plt.scatter(X, y, color = 'red')
plt.plot(X, lin_reg_2.predict(poly_reg.fit_transform(X)), color = 'blue')
plt.title('Truth or Bluff (Polynomial Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```



In [13]:
```python
# Visualising the Polynomial Regression results (for higher resolution and smoother curve)
X_grid = np.arange(min(X), max(X), 0.1)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X, y, color = 'red')
plt.plot(X_grid, lin_reg_2.predict(poly_reg.fit_transform(X_grid)), color = 'blue')
plt.title('Truth or Bluff (Polynomial Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```



Support Vector Regression (SVR)

In [14]:
```python
# Fitting SVR to the dataset
from sklearn.svm import SVR
regressor = SVR(kernel = 'rbf')
# regressor = SVR(kernel="linear", C=1.0, epsilon=0.2)
regressor.fit(X, y)
```

Out[14]:
```
▾ SVR
SVR()
```

In [15]:
```python
# Predicting a new result
y_pred = regressor.predict([[6.5]])
y_pred
```

Out[15]:
```
array([130001.82883924])
```

In [16]:
```python
y_pred = regressor.predict(X_test)
y_pred
```

Out[16]:
```
array([129996.54009563, 130003.51550835])
```

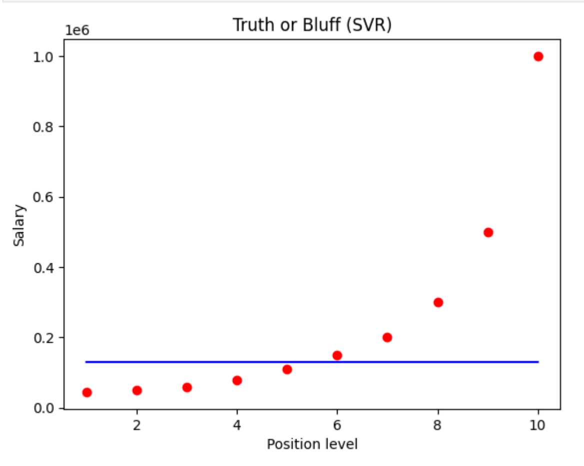
In [17]:
```python
print("DATOS DEL MODELO VECTORES DE SOPORTE REGRESIÓN")
print()

print("Precisión del modelo:")
print(regressor.score(X_train, y_train))
```

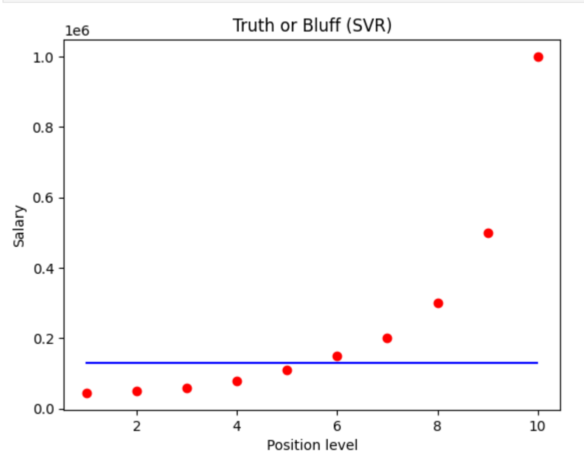
```
DATOS DEL MODELO VECTORES DE SOPORTE REGRESIÓN

Precisión del modelo:
-0.1415131652565642
```

In [18]:
```python
# 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()
```



In [19]:
```python
# 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 scale
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()
```



Decision Tree Regression

In [20]:
```python
# Fitting Decision Tree Regression to the dataset
from sklearn.tree import DecisionTreeRegressor

regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(X, y)
```

Out[20]:
```
▾ DecisionTreeRegressor
DecisionTreeRegressor(random_state=0)
```

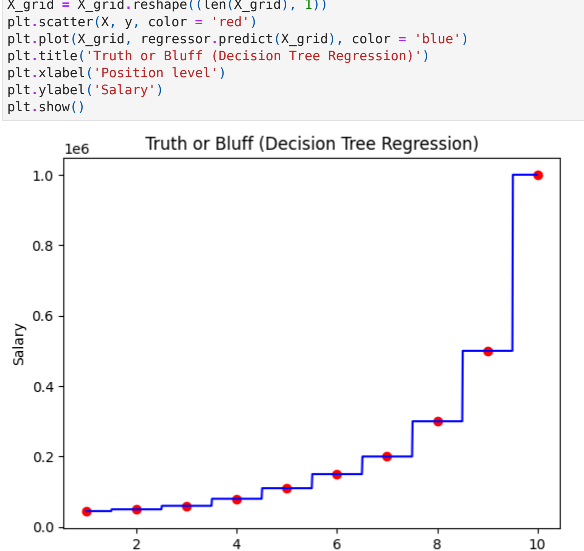
In [22]:
```python
# Predicting a new result
y_pred = regressor.predict([[6.5]])
y_pred
```

Out[22]:
```
array([150000.])
```

In [23]:
```python
y_pred = regressor.predict(X_test)
y_pred
```

Out[23]:
```
array([ 60000., 500000.])
```

In [24]:
```python
# Visualising the Decision Tree Regression results (higher resolution)
X_grid = np.arange(min(X), max(X), 0.01)
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 (Decision Tree Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```



In [26]:
```python
print("Datos del Modelo de árboles de decision Regresión")
print()

print("Precisión del modelo:")
print(regressor.score(X_train, y_train))
```

```
Datos del Modelo de árboles de decision Regresión

Precisión del modelo:
1.0
```

Random Forest Regression

In [27]:
```python
# Fitting Random Forest Regression to the dataset
from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
regressor.fit(X, y)
```

Out[27]:
```
▾ RandomForestRegressor
RandomForestRegressor(n_estimators=10, random_state=0)
```

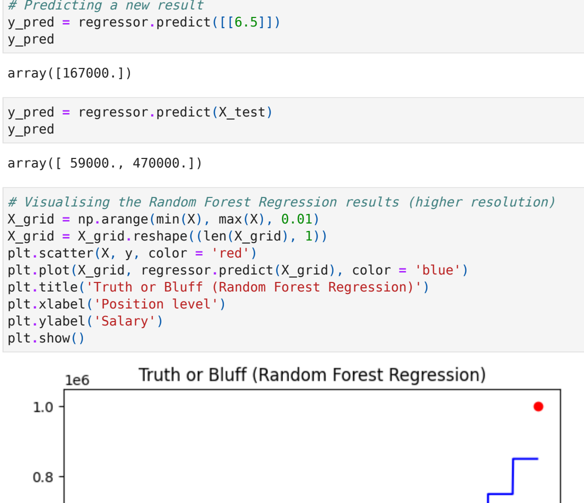
In [28]:
```python
# Predicting a new result
y_pred = regressor.predict([[6.5]])
y_pred
```

Out[28]:
```
array([167000.])
```

In [29]:
```python
y_pred = regressor.predict(X_test)
y_pred
```

Out[29]:
```
array([ 59000., 470000.])
```

In [30]:
```python
# Visualising the Random Forest Regression results (higher resolution)
X_grid = np.arange(min(X), max(X), 0.01)
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 (Random Forest Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```



In [32]:
```python
print("Datos del modelo Bosques Aleatorios Regresión")
print()

print("precisión del modelo:")
print(regressor.score(X_train, y_train))
```

```
Datos del modelo Bosques Aleatorios Regresión

precisión del modelo:
0.9675758285151437
```

Conclusión

Los dos mejores modelos son el Polinomial y el Random Forest, como la precisión es más alta nos quedaremos con Polinomial

Creado por:

Isabel Maniega