C:\VS Code\Spyder\Comparison of all regressions for emp salary.py

```
# -------------- Comparing all Regression Models ------------------- #

#importing libraries

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

#importing  the dataset

dataset = pd.read_csv(r"C:\Users\Jan Saida\Downloads\emp_sal.csv")
x=dataset.iloc[:, 1:2].values  #independent varianble
y=dataset.iloc[:,2].values     #dependent varianble

#Linear reg model -- linear algorithm (bydefault degree - 1)

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


#Linear regression Predictions

lin_model_pred=lin_reg.predict([[6.5]])
lin_model_pred

#Linear regression visualization

plt.scatter(x,y,color='red')
plt.plot(x,lin_reg.predict(x),color='blue')
plt.title('Linear Regression graph')
plt.xlabel('Position level')
plt.ylabel('Salary')

#polynomial  regression model (bydefault degree - 2)

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

poly_reg.fit(X_poly,y)

lin_reg_2=LinearRegression()

lin_reg_2.fit(X_poly,y)

#Polynomial regression Predictions

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

```
#polynomial visualisation

plt.scatter(x,y,color='red')
plt.plot(x,lin_reg_2.predict(poly_reg.fit_transform(x)),color='blue')
plt.title('Polynomial Regression graph')
plt.xlabel('Position level')
plt.ylabel('Salary')

#support vector regression model (degree=5)

from sklearn.svm import SVR
svr_reg=SVR(kernel='poly', gamma='scale', degree=5,C=1)
svr_reg.fit(x,y)

#Support vector regression Predictions

svr_reg_pred=svr_reg.predict([[6.5]])
svr_reg_pred


# Support Vector regressor Visualizations

plt.scatter(x,y,color='red')
plt.plot(x,svr_reg.predict(x),color='blue')
plt.title('Simple Vector Regression graph')
plt.xlabel('Position level')
plt.ylabel('Salary')

# knn regression model

from sklearn.neighbors import KNeighborsRegressor
knn_reg = KNeighborsRegressor(n_neighbors=4,weights='distance')
knn_reg.fit(x,y)

#knn regression Predictions

knn_reg_pred=knn_reg.predict([[6.5]])
knn_reg_pred

# Knn regressor Visualizations

plt.scatter(x,y,color='red')
plt.plot(x,knn_reg.predict(x),color='blue')
plt.title('Knn Regression graph')
plt.xlabel('Position level')
plt.ylabel('Salary')


#desicion tree algorithm

from sklearn.tree import DecisionTreeRegressor
tree_reg = DecisionTreeRegressor( splitter="best", random_state=0)
tree_reg.fit(x,y)
```

```
#descision tree Predictions

tree_reg_pred=tree_reg.predict([[6.5]])
tree_reg_pred

# Desicion Tree regressor Visualizations

plt.scatter(x,y,color='red')
plt.plot(x,tree_reg.predict(x),color='blue')
plt.title('Decision Tree Regression graph')
plt.xlabel('Position level')
plt.ylabel('Salary')

# Random foreest Regression model

from sklearn.ensemble import RandomForestRegressor
forest_regressor=RandomForestRegressor(n_estimators=15)
forest_regressor.fit(x,y)

# Random Forest regressor Predictions

forest_pred=forest_regressor.predict([[6.5]])
forest_pred

# Random Forest Regressor Visualizations

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,forest_regressor.predict(x_grid),color='blue')
plt.title('Random Forest Regression graph')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()

# Extreme Gradiant Boosting Regressor Model

import xgboost as xg
xgb_r = xg.XGBRegressor(objective = 'reg:linear', n_estimators = 10, seed = 0)
xgb_r.fit(x,y)

# Extreme Gradiant Boosting Model Predictions

xgb_reg_pred = xgb_r.predict([[6.5]])
xgb_reg_pred



# Pickle Linear Regression Model
with open('linear_regression_model.pkl', 'wb') as file:
    pickle.dump(lin_reg, file)

# Pickle Polynomial Regression Model
```

```
with open('polynomial_regression_model.pkl', 'wb') as file:
    pickle.dump(lin_reg_2, file)

# Pickle Support Vector Regression Model
with open('svr_model.pkl', 'wb') as file:
    pickle.dump(svr_reg, file)

# Pickle KNN Regression Model
with open('knn_model.pkl', 'wb') as file:
    pickle.dump(knn_reg, file)

# Pickle Decision Tree Model
with open('decision_tree_model.pkl', 'wb') as file:
    pickle.dump(tree_reg, file)

# Pickle Random Forest Model
with open('random_forest_model.pkl', 'wb') as file:
    pickle.dump(forest_regressor, file)

# Pickle Extreme Gradient Boosting Model
with open('xgboost_model.pkl', 'wb') as file:
    pickle.dump(xgb_r, file)


```

Streamlit\Comparison_of_all_regs_emp_salary.py

```python
import streamlit as st
import pickle
import numpy as np

# Load the pickled models
with open(r'C:\Users\Jan Saida\linear_regression_model.pkl', 'rb') as file:
    lin_reg = pickle.load(file)

with open(r'C:\Users\Jan Saida\polynomial_regression_model.pkl', 'rb') as file:
    lin_reg_2 = pickle.load(file)

with open(r'C:\Users\Jan Saida\svr_model.pkl', 'rb') as file:
    svr_reg = pickle.load(file)

with open(r'C:\Users\Jan Saida\knn_model.pkl', 'rb') as file:
    knn_reg = pickle.load(file)

with open(r'C:\Users\Jan Saida\decision_tree_model.pkl', 'rb') as file:
    tree_reg = pickle.load(file)

with open(r'C:\Users\Jan Saida\random_forest_model.pkl', 'rb') as file:
    forest_regressor = pickle.load(file)

with open(r'C:\Users\Jan Saida\xgboost_model.pkl', 'rb') as file:
    xgb_r = pickle.load(file)

# Polynomial Features (this is the same across different models)
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=6)

# Streamlit UI
st.title('Salary Prediction using Multiple Regression Models')

# Input for Position level
position_level = st.number_input('Enter the position level (e.g., 6.5):', min_value=1.0,
max_value=10.0, value=6.5)

# Polynomial Regression Prediction
position_level_poly = poly_reg.fit_transform([[position_level]])  # Apply transformation
poly_pred = lin_reg_2.predict(position_level_poly)  # Predict using the transformed features

# Linear Regression Prediction
lin_pred = lin_reg.predict([[position_level]])

# Support Vector Regression Prediction
svr_pred = svr_reg.predict([[position_level]])

# KNN Regression Prediction
knn_pred = knn_reg.predict([[position_level,0]])

# Decision Tree Prediction
tree_pred = tree_reg.predict([[position_level]])
```

```
52
53  # Random Forest Prediction
54  forest_pred = forest_regressor.predict([[position_level]])
55
56  # XGBoost Prediction
57  xgb_pred = xgb_r.predict([[position_level]])
58
59  # Display the results
60  st.subheader(f'Predicted Salary for Position Level {position_level}')
61  st.write(f"Linear Regression Prediction: {lin_pred[0]}")
62  st.write(f"Polynomial Regression Prediction: {poly_pred[0]}")
63  st.write(f"Support Vector Regression Prediction: {svr_pred[0]}")
64  st.write(f"KNN Regression Prediction: {knn_pred[0]}")
65  st.write(f"Decision Tree Prediction: {tree_pred[0]}")
66  st.write(f"Random Forest Prediction: {forest_pred[0]}")
67  st.write(f"XGBoost Prediction: {xgb_pred[0]}")
68
```

Salary Prediction using Multiple Regression Models

Enter the position level (e.g., 6.5):

6.50 − +

Predicted Salary for Position Level 6.5

Linear Regression Prediction: 330378.78787878784

Polynomial Regression Prediction: 174192.81930661248

Support Vector Regression Prediction: 164079.01344549266

KNN Regression Prediction: 1

Decision Tree Prediction: 150000.0

Random Forest Prediction: 157333.33333333334

XGBoost Prediction: 158778.203125

Salary Prediction using Multiple Regression Models

Enter the position level (e.g., 6.5):

2.20 − +

Predicted Salary for Position Level 2.2

Linear Regression Prediction: -17400.0

Polynomial Regression Prediction: 51187.390522576505

Support Vector Regression Prediction: 100009.9473626854

KNN Regression Prediction: 1

Decision Tree Prediction: 50000.0

Random Forest Prediction: 55333.333333333336

XGBoost Prediction: 68077.109375