Project on Salary Dataset

```
#Improting Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression,LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import r2_score
```

```
#load salary dataset
df=pd.read_csv("Salary_Data.csv")
df
```

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0
5	2.9	56642.0
6	3.0	60150.0
7	3.2	54445.0
8	3.2	64445.0
9	3.7	57189.0
10	3.9	63218.0
11	4.0	55794.0
12	4.0	56957.0
13	4.1	57081.0
14	4.5	61111.0
15	4.9	67938.0
16	5.1	66029.0
17	5.3	83088.0
18	5.9	81363.0
19	6.0	93940.0
20	6.8	91738.0
21	7.1	98273.0
22	7.9	101302.0
23	8.2	113812.0
24	8.7	109431.0
25	9.0	105582.0
26	9.5	116969.0
27	9.6	112635.0
28	10.3	122391.0
29	10.5	121872.0

```
#show first five Rows
df.head()
```

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

```
#Information of data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     float64
dtypes: float64(2)
memory usage: 612.0 bytes
```

```
#Analysing the Dataset
df.describe()
```

	YearsExperience	Salary
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785
min	1.100000	37731.000000
25%	3.200000	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

```
#Checking Nullvalue
df.isnull().sum()
```

	0
YearsExperience	0
Salary	0

dtype: int64

```
#split fetures and target
x=df[['YearsExperience']]
y=df['Salary']
```

```
#Train Test Split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

using Liner Regression Algorithm

```
linear=LinearRegression()
linear.fit(x_train,y_train)

y_pred=linear.predict(x_test)
print("Linear Regression Prediction:", y_pred)
print("Linear Regression Accuracy:",linear.score(x_test,y_test))


#Predict salary for 5 years experience
print("Salary for 5 years:",linear.predict([[5]]))
```

```
Linear Regression Prediction: [ 92898.14624275  36584.05415965  61828.30233483  53089.90873573
  62799.23495695 125909.85539491]
Linear Regression Accuracy: 0.9473833071419254
Salary for 5 years: [72508.56117818]
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names,
  warnings.warn(
```

Using DecisionTree Algorithm

```
decision=DecisionTreeClassifier()
decision.fit(x_train,y_train)

y_pred=decision.predict(x_test)
print("Decision Tree Accuracy:",accuracy_score(y_test,y_pred))

print("Salary class for 5 years:",decision.predict([[5]]))
```

```
Decision Tree Accuracy: 0.0
Salary class for 5 years: [67938.]
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names,
  warnings.warn(
```

Using logistic regression Algorithm

```


y_class=(y>60000).astype(int)

x_trian,x_test,y_train,y_test=train_test_split(x,y_class)

logistic=LogisticRegression()
logistic.fit(x_trian,y_train)

y_pred=logistic.predict(x_test)
print("Logistic Regression Accuracy:",accuracy_score(y_test,y_pred))

print("Salary class for 5 years:",logistic.predict([[5]]))
```

```
Logistic Regression Accuracy: 0.75
Salary class for 5 years: [1]
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names,
  warnings.warn(
```

```
test_experience = [[10]]

predicted_salary = linear.predict(test_experience)

print("Predicted Salary for 5 years experience:", predicted_salary[0])
```

```
Predicted Salary for 5 years experience: 121055.19228429723
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names,
  warnings.warn(
```

```
acc_lr = 0.6728365409861852  # Linear Regression
acc_dt = 0.0                 # Decision Tree
```

```
acc_logr = 0.75             # Logistic Regression

# Note: Direct comparison of these metrics might be misleading as they represent different

if acc_lr > acc_logr and acc_lr > acc_dt:
  print("Best Algorithm: Linear Regression")
elif acc_logr > acc_lr and acc_logr > acc_dt:
  print("Best Algorithm: Logistic Regression")
else:
  print("Best Algorithm: Decision Tree")
```

```
Best Algorithm: Logistic Regression
```