In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import LabelEncoder,OneHotEncoder,StandardScaler
import warnings
warnings.filterwarnings('ignore')
from sklearn.tree import DecisionTreeRegressor
from sklearn.feature_selection import RFE
from sklearn.preprocessing import MinMaxScaler,StandardScaler
import joblib
import datetime
from sklearn.feature_selection import SelectFromModel
from sklearn.model_selection import cross_val_score,GridSearchCV
from sklearn.linear_model import Ridge,Lasso
```

In [2]:
```
df= pd.read_csv('HR.csv')
```

In [3]:
```
df
```

Out[3]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	E
0	41	Yes	Travel_Rarely	1102	Sales	1	2	
1	49	No	Travel_Frequently	279	Research & Development	8	1	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	
4	27	No	Travel_Rarely	591	Research & Development	2	1	
...	...	...	...	...	...	...	...	...
1465	36	No	Travel_Frequently	884	Research & Development	23	2	
1466	39	No	Travel_Rarely	613	Research & Development	6	1	
1467	27	No	Travel_Rarely	155	Research & Development	4	3	
1468	49	No	Travel_Frequently	1023	Sales	2	3	
1469	34	No	Travel_Rarely	628	Research & Development	8	3	

1470 rows × 35 columns

In [4]:
```
df.head()
```

Out[4]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educ
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life
4	27	No	Travel_Rarely	591	Research & Development	2	1	

5 rows × 35 columns

In [5]:
```
df.tail()
```

Out[5]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	E
1465	36	No	Travel_Frequently	884	Research & Development	23	2	
1466	39	No	Travel_Rarely	613	Research & Development	6	1	
1467	27	No	Travel_Rarely	155	Research & Development	4	3	
1468	49	No	Travel_Frequently	1023	Sales	2	3	
1469	34	No	Travel_Rarely	628	Research & Development	8	3	

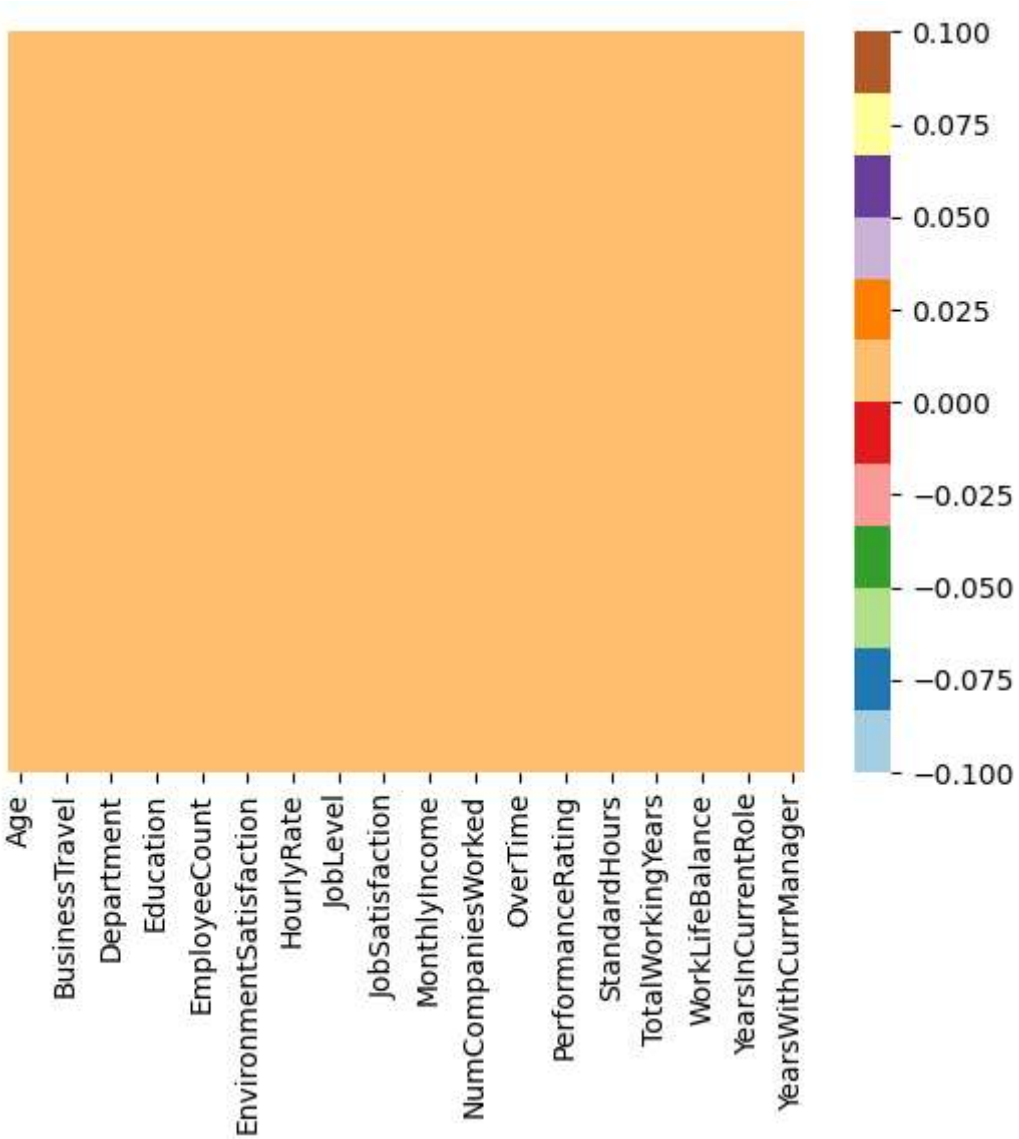
5 rows × 35 columns

In [6]:
```
df.isnull().sum()
```

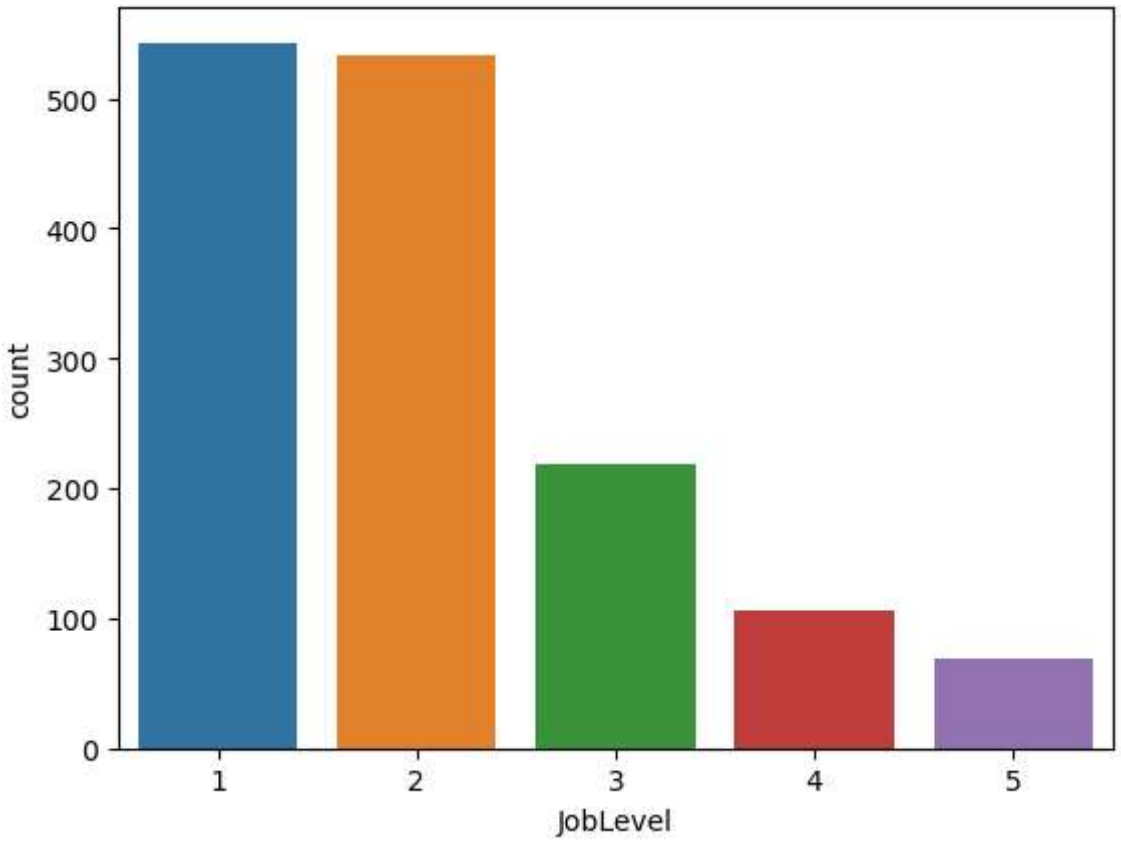
Out[6]:
```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

In [7]:
```
sns.heatmap(df.isnull(),yticklabels=False,cmap="Paired")
```

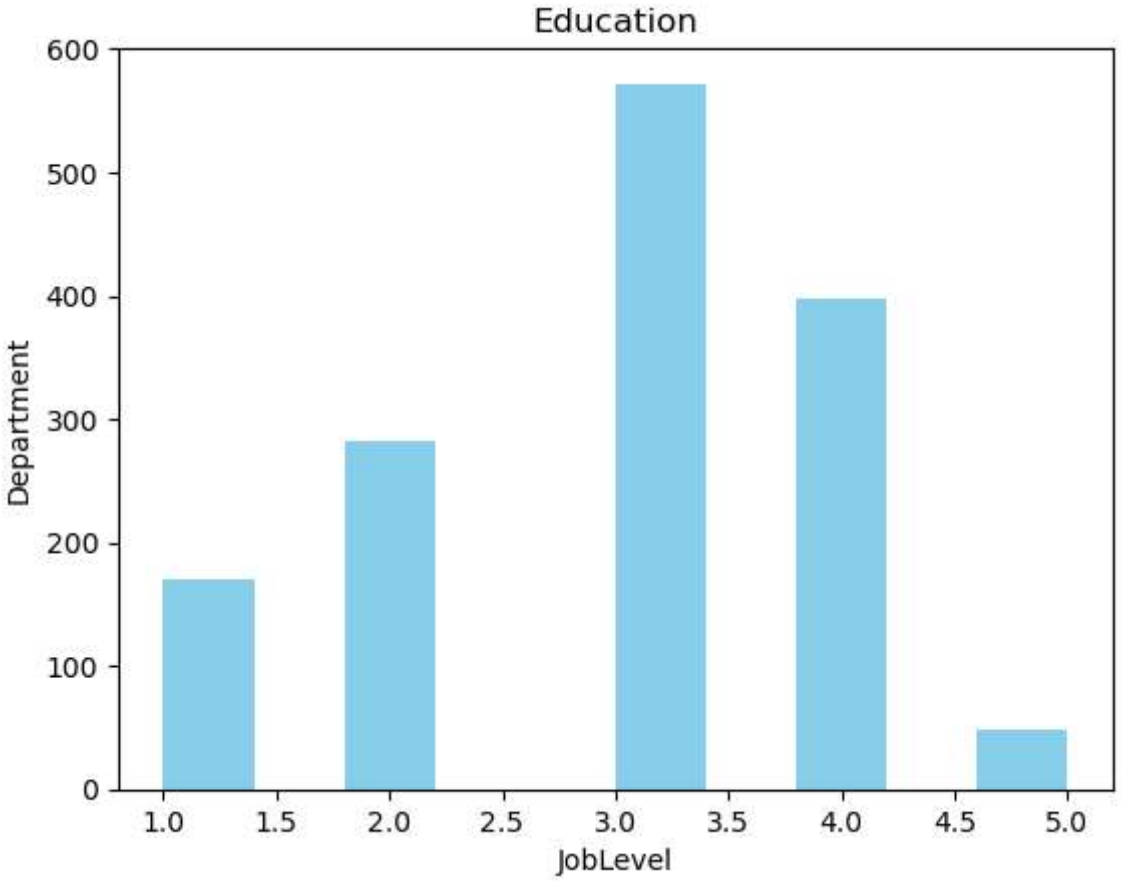
Out[7]: <Axes: >



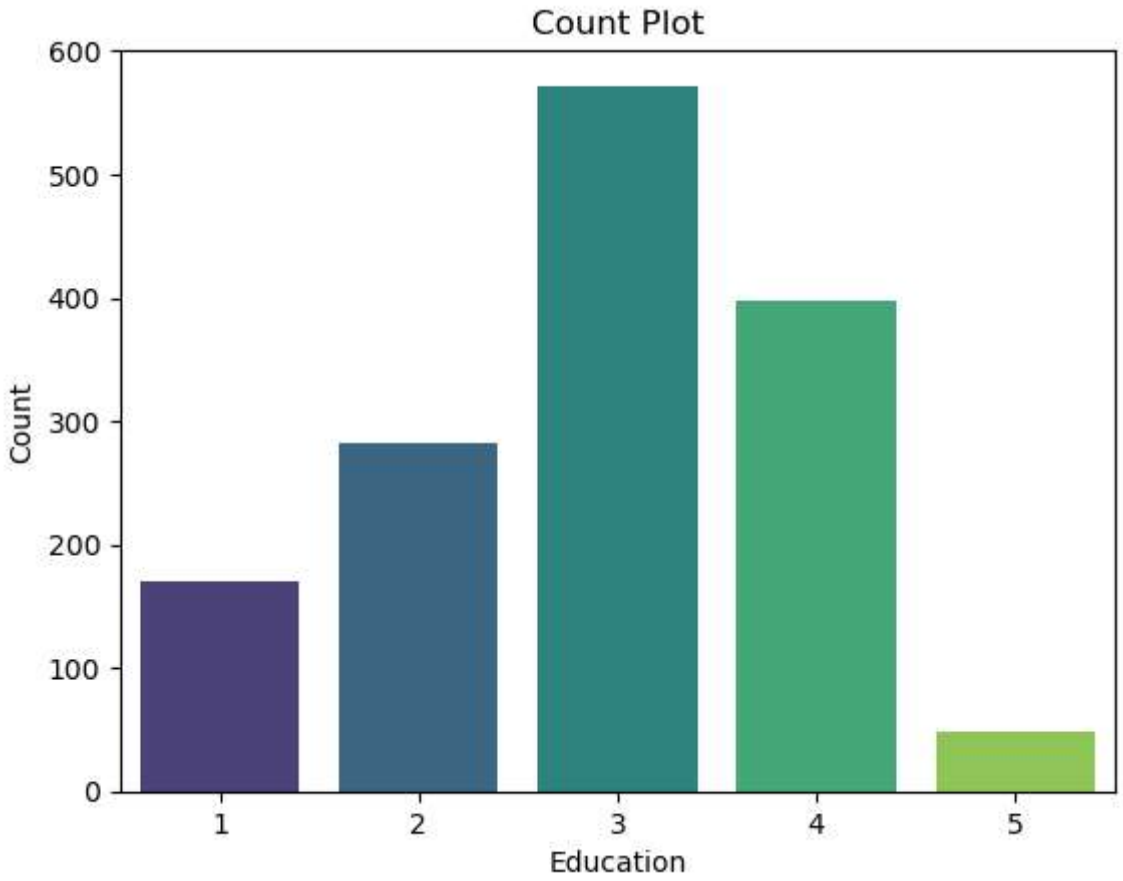
In [8]:
```
sns.countplot(x=df['JobLevel'])
plt.show()
```



In [9]:
```
plt.hist(df['Education'], bins=10, color='skyblue')

plt.xlabel('JobLevel')
plt.ylabel('Department')
plt.title('Education')
plt.show()
```



In [10]:
```
sns.countplot(x='Education', data=df, palette='viridis')
plt.xlabel('Education')
plt.ylabel('Count')
plt.title('Count Plot')

plt.show()
```

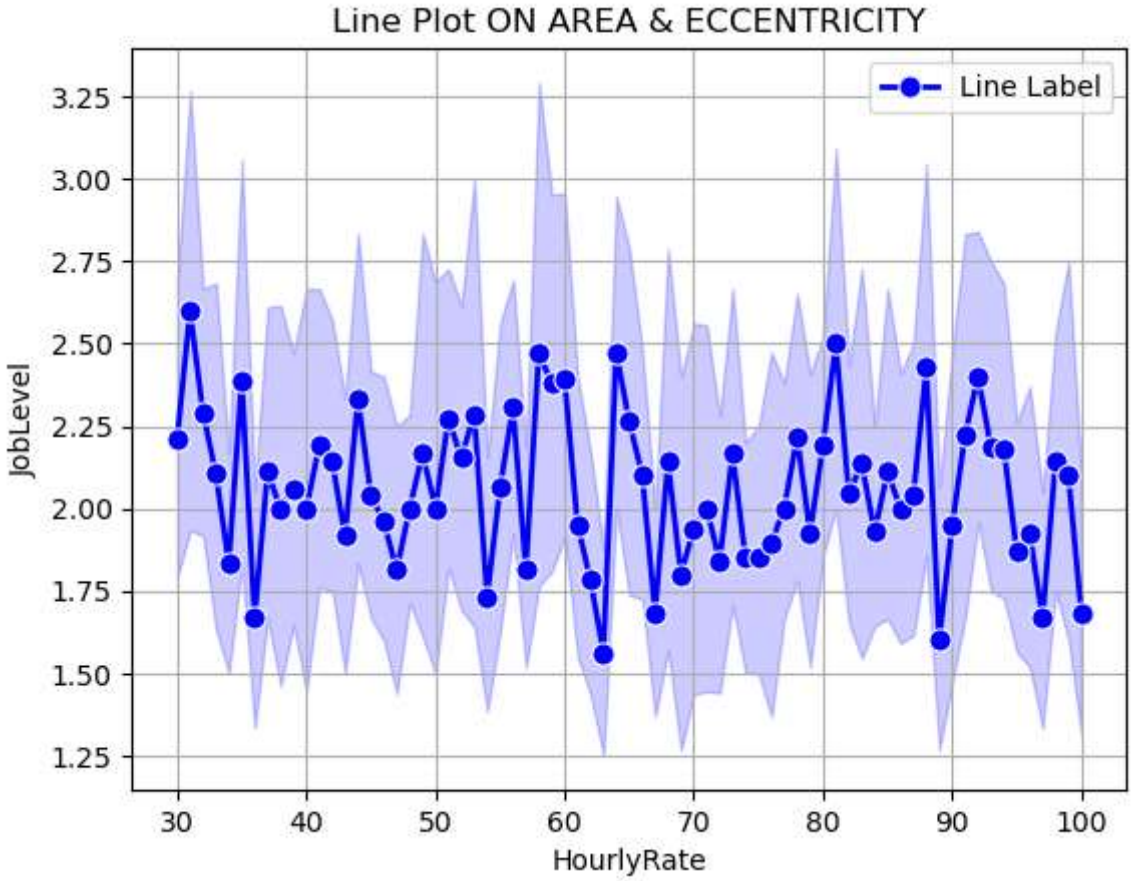


In [11]:
```
df_area=df['HourlyRate'].value_counts()
```

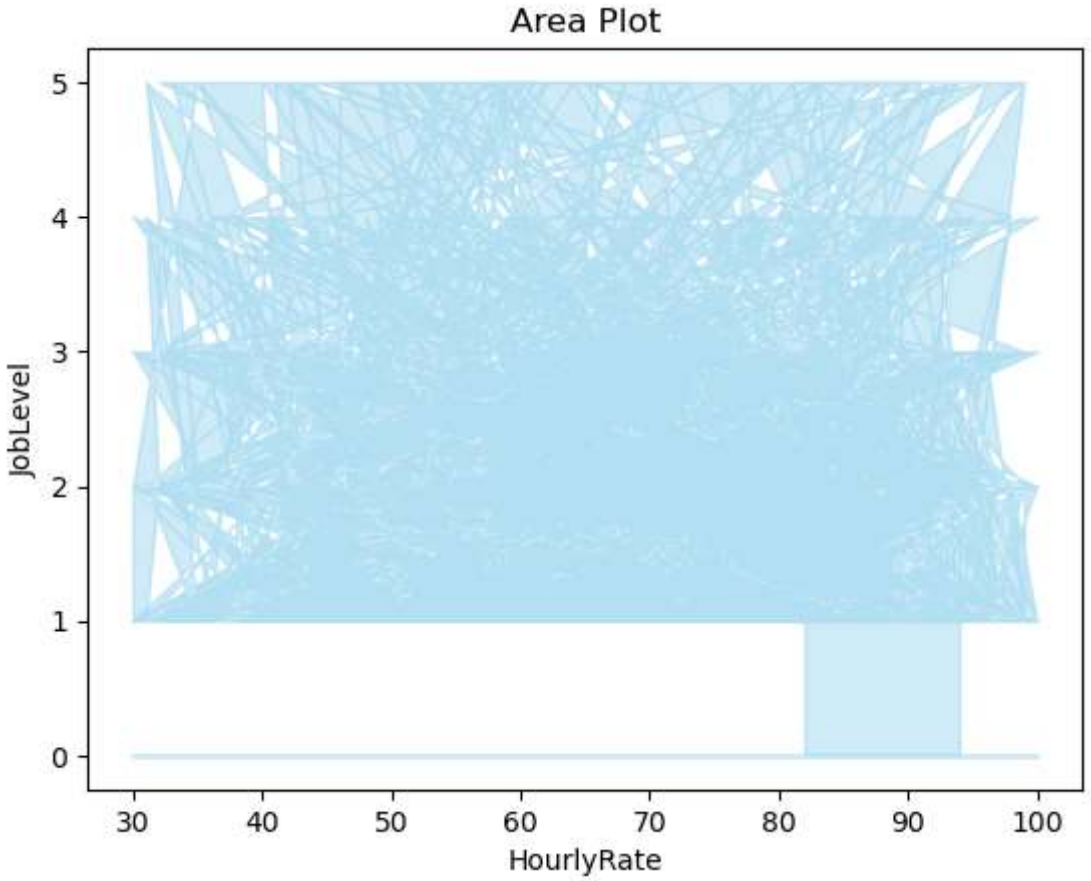
In [12]:
```
df_area
```

Out[12]:
```
66    29
98    28
42    28
48    28
84    28
      ..
31    15
53    14
68    14
38    13
34    12
Name: HourlyRate, Length: 71, dtype: int64
```

In [13]:
```
sns.lineplot(x='HourlyRate', y='JobLevel', data=df, marker='o', color='b', lin

plt.xlabel('HourlyRate')
plt.ylabel('JobLevel')
plt.title('Line Plot ON AREA & ECCENTRICITY')
plt.legend()
plt.grid(True)
plt.show()
```



In [14]:
```
plt.fill_between(df['HourlyRate'], df['JobLevel'], color='skyblue', alpha=0.4)
plt.xlabel('HourlyRate')
plt.ylabel('JobLevel')
plt.title('Area Plot')
plt.show()
```

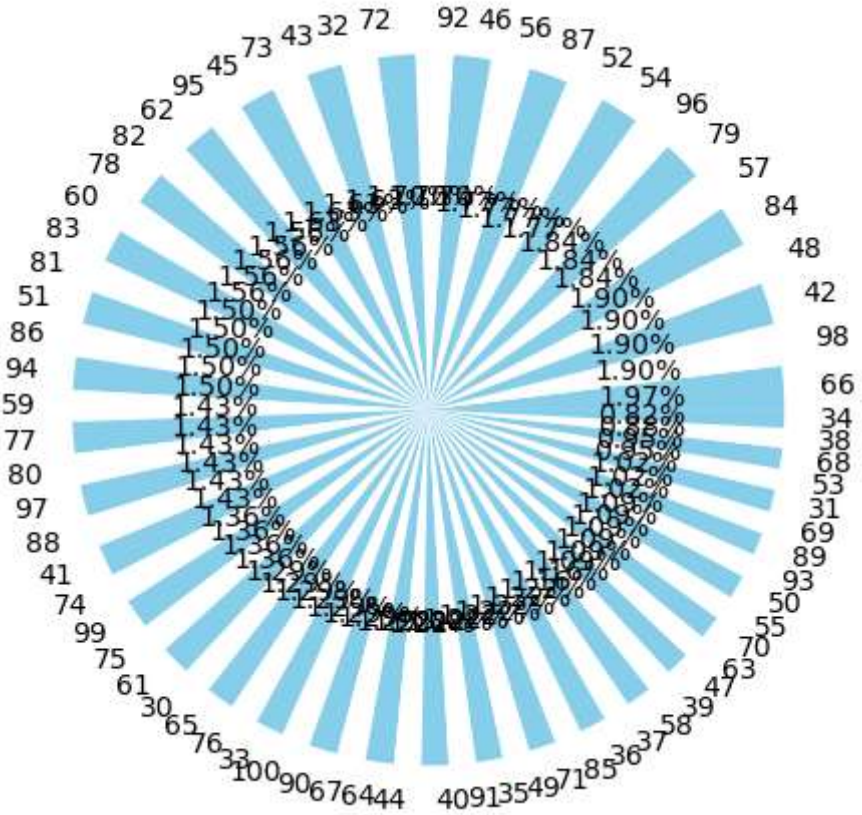


In [15]:
```
df_pie =df['HourlyRate'].value_counts()
```

In [16]:
```
df_pie
```

Out[16]:
```
66    29
98    28
42    28
48    28
84    28
      ..
31    15
53    14
68    14
38    13
34    12
Name: HourlyRate, Length: 71, dtype: int64
```

In [17]:
```
plt.pie(df_pie,labels=df_pie.index,autopct="%.2f%%",radius =1.2,colors=['skybl

plt.show()
```



In [18]:
```
df.dtypes
```

Out[18]:
```
Age                         int64
Attrition                  object
BusinessTravel             object
DailyRate                   int64
Department                 object
DistanceFromHome            int64
Education                   int64
EducationField             object
EmployeeCount               int64
EmployeeNumber              int64
EnvironmentSatisfaction     int64
Gender                     object
HourlyRate                  int64
JobInvolvement              int64
JobLevel                    int64
JobRole                    object
JobSatisfaction             int64
MaritalStatus              object
MonthlyIncome               int64
MonthlyRate                 int64
NumCompaniesWorked          int64
Over18                     object
OverTime                   object
PercentSalaryHike           int64
PerformanceRating           int64
RelationshipSatisfaction    int64
StandardHours               int64
StockOptionLevel            int64
TotalWorkingYears           int64
TrainingTimesLastYear       int64
WorkLifeBalance             int64
YearsAtCompany              int64
YearsInCurrentRole          int64
YearsSinceLastPromotion     int64
YearsWithCurrManager        int64
dtype: object
```

In [19]:
```
df
```

Out[19]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	E
0	41	Yes	Travel_Rarely	1102	Sales	1	2	
1	49	No	Travel_Frequently	279	Research & Development	8	1	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	
4	27	No	Travel_Rarely	591	Research & Development	2	1	
...	...	...	...	...	...	...	...	
1465	36	No	Travel_Frequently	884	Research & Development	23	2	
1466	39	No	Travel_Rarely	613	Research & Development	6	1	
1467	27	No	Travel_Rarely	155	Research & Development	4	3	
1468	49	No	Travel_Frequently	1023	Sales	2	3	
1469	34	No	Travel_Rarely	628	Research & Development	8	3	

1470 rows × 35 columns

ENCODING

In [20]:
```
encoder=LabelEncoder()
```

In [21]:
```
categorical_col = ['Attrition','BusinessTravel','Department','EducationField',
encoder = OneHotEncoder(drop='first',sparse=False)
encoder_cols = pd.DataFrame(encoder.fit_transform(df[categorical_col]),columns
```

In [22]:
```
encoder_cols
```

Out[22]:

	Attrition_Yes	BusinessTravel_Travel_Frequently	BusinessTravel_Travel_Rarely	Department_I & Dev
0	1.0	0.0	1.0	
1	0.0	1.0	0.0	
2	1.0	0.0	1.0	
3	0.0	1.0	0.0	
4	0.0	0.0	1.0	
...	...	...	...	
1465	0.0	1.0	0.0	
1466	0.0	0.0	1.0	
1467	0.0	0.0	1.0	
1468	0.0	1.0	0.0	
1469	0.0	0.0	1.0	

1470 rows × 22 columns

In [23]:
```
numerical_col = ['Age','DailyRate','DistanceFromHome','Education','EmployeeCou
Scaled = StandardScaler()
Scaled= pd.DataFrame(Scaled.fit_transform(df[numerical_col]),columns=numerical
```

In [24]:
```
Scaled
```

Out[24]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	R
0	0.446350	0.742527	-1.010909	-0.891688	0.0	-1.701283	
1	1.322365	-1.297775	-0.147150	-1.868426	0.0	-1.699621	
2	0.008343	1.414363	-0.887515	-0.891688	0.0	-1.696298	
3	-0.429664	1.461466	-0.764121	1.061787	0.0	-1.694636	
4	-1.086676	-0.524295	-0.887515	-1.868426	0.0	-1.691313	
...	...	...	...	...	...	...	
1465	-0.101159	0.202082	1.703764	-0.891688	0.0	1.721670	
1466	0.227347	-0.469754	-0.393938	-1.868426	0.0	1.723332	
1467	-1.086676	-1.605183	-0.640727	0.085049	0.0	1.726655	
1468	1.322365	0.546677	-0.887515	0.085049	0.0	1.728317	
1469	-0.320163	-0.432568	-0.147150	0.085049	0.0	1.733302	

1470 rows × 15 columns

In [25]:
```
x = pd.concat([encoder_cols,Scaled],axis=1)
y = df['YearsWithCurrManager']
```

In [26]:
```
x
```

Out[26]:

	Attrition_Yes	BusinessTravel_Travel_Frequently	BusinessTravel_Travel_Rarely	Department_I & Dev
0	1.0	0.0	1.0	
1	0.0	1.0	0.0	
2	1.0	0.0	1.0	
3	0.0	1.0	0.0	
4	0.0	0.0	1.0	
...	...	...	...	
1465	0.0	1.0	0.0	
1466	0.0	0.0	1.0	
1467	0.0	0.0	1.0	
1468	0.0	1.0	0.0	
1469	0.0	0.0	1.0	

1470 rows × 37 columns

In [27]:
```
y
```

Out[27]:
```
0       5
1       7
2       0
3       0
4       2
       ..
1465    3
1466    7
1467    3
1468    8
1469    2
Name: YearsWithCurrManager, Length: 1470, dtype: int64
```

LINEAR

In [28]:
```
x.shape
```

Out[28]:
```
(1470, 37)
```

In [29]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=
```

In [30]:
```
x_train
```

Out[30]:

	Attrition_Yes	BusinessTravel_Travel_Frequently	BusinessTravel_Travel_Rarely	Department_ & Dev
1097	0.0	0.0	1.0	
727	0.0	0.0	0.0	
254	0.0	0.0	1.0	
1175	0.0	0.0	1.0	
1341	0.0	0.0	1.0	
...	...	...	...	
1130	0.0	0.0	1.0	
1294	0.0	0.0	1.0	
860	1.0	1.0	0.0	
1459	0.0	0.0	1.0	
1126	0.0	0.0	1.0	

1176 rows × 37 columns

In [31]:
```
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
```

In [32]:
```
from sklearn.metrics import r2_score, mean_squared_error,mean_absolute_error
```

In [33]:
```
R2= r2_score(y_test,y_pred)
```

In [34]:
```
mae = mean_absolute_error(y_test,y_pred)
mse = mean_squared_error(y_test,y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test,y_pred)
```

In [35]:
```
print('Mean Absolute Error',mae)
print('Mean Squared error',rmse)
print('Root Mean Squared Error',rmse)
print('R2 Score',r2)
```

```
Mean Absolute Error 1.557335466097933
Mean Squared error 2.0962789390706984
Root Mean Squared Error 2.0962789390706984
R2 Score 0.6798528904675876
```

In [36]:
```
lr_model = LinearRegression()
lr_scores = cross_val_score(lr_model,x_train,y_train,cv=5)
```

In [37]:
```
ridge_model= Ridge(alpha=1.0)
ridge_scores = cross_val_score(ridge_model , x_train , y_train , cv = 5)
```

In [38]:
```
Lasso_model= Lasso(alpha=1.0)
Lasso_scores = cross_val_score(Lasso_model , x_train , y_train , cv = 5)
```

In [39]:
```
lr_model.fit(x_train,y_train)
lr_prediction = lr_model.predict(x_test)
lr_mae = mean_absolute_error(y_test,lr_prediction)
lr_mse = mean_squared_error(y_test,lr_prediction)
lr_r2 = r2_score(y_test,lr_prediction)
```

In [40]:
```
print('Linear MAE',lr_mae)
print('Linear MSE',lr_mse)
print('Linear R2',lr_r2)
```

```
Linear MAE 1.557335466097933
Linear MSE 4.394385390391372
Linear R2 0.6798528904675876
```

In [41]:
```
Lasso_model.fit(x_train , y_train)
Lasso_prediction = Lasso_model.predict(x_test)
Lasso_mae = mean_absolute_error(y_test ,Lasso_prediction)
Lasso_mse = mean_squared_error(y_test ,Lasso_prediction)
Lasso_r2 = r2_score(y_test ,Lasso_prediction)
```

In [42]:
```
print('Lasso MAE',Lasso_mae)
print('Lasso MSE',Lasso_mse)
print('Lasso R2',Lasso_r2)
```

```
Lasso MAE 2.0235815126568895
Lasso MSE 6.188584184887578
Lasso R2 0.5491389209462796
```

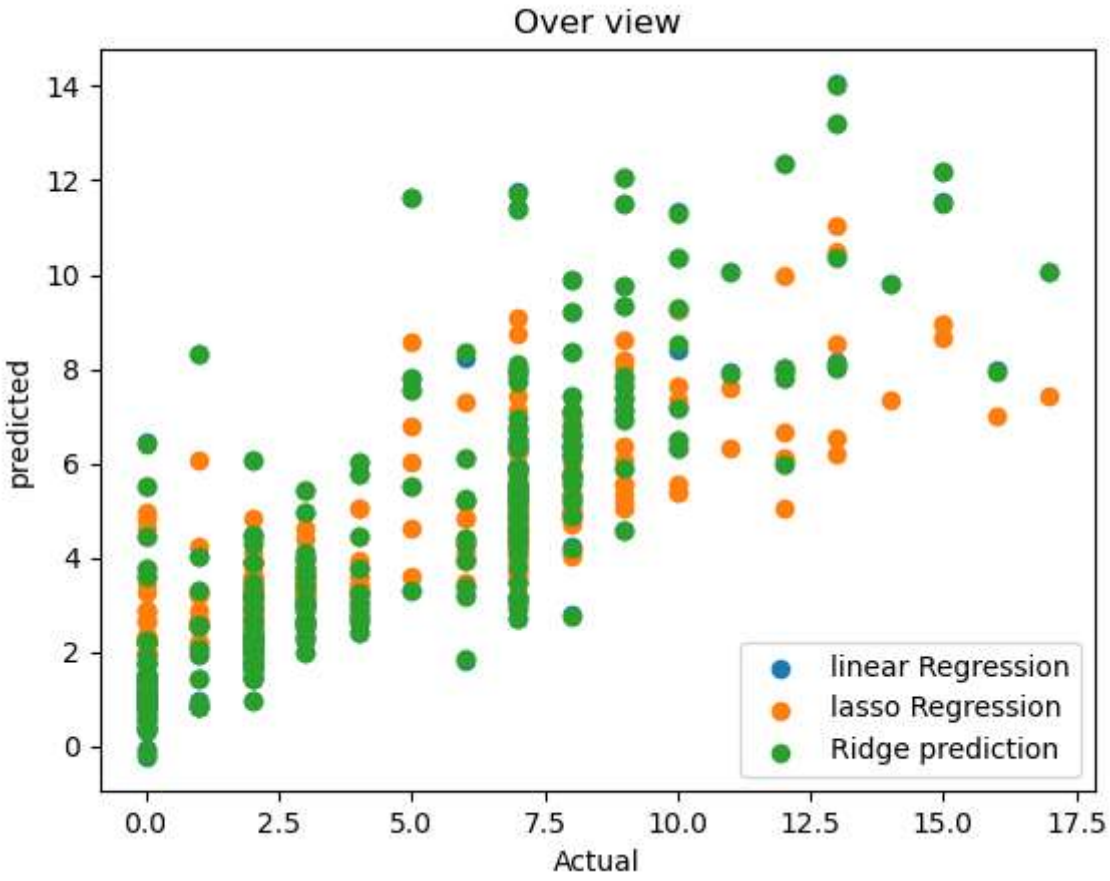
In [43]:
```
ridge_model.fit(x_train , y_train)
ridge_prediction = ridge_model.predict(x_test)
ridge_mae = mean_absolute_error(y_test ,ridge_prediction)
ridge_mse = mean_squared_error(y_test ,ridge_prediction)
ridge_r2 = r2_score(y_test ,ridge_prediction)
```

In [44]:
```
print('Ridge MAE',ridge_mae)
print('Ridge MSE',ridge_mse)
print('Ridge R2',ridge_r2)
```

```
Ridge MAE 1.5567969339805532
Ridge MSE 4.394584864652236
Ridge R2 0.6798383580353204
```

In [45]:
```
plt.scatter(y_test,lr_prediction,alpha = 1.0,label='linear Regression')
plt.scatter(y_test,Lasso_prediction,alpha=1.0,label= 'lasso Regression')
plt.scatter(y_test,ridge_prediction,alpha=1.0,label='Ridge prediction')

plt.xlabel('Actual')
plt.ylabel('predicted')
plt.title('Over view')
plt.legend()
```

Out[45]:
```
<matplotlib.legend.Legend at 0x199b6381250>
```



LOGISTIC REGRESSION

In [46]:
```
def YearsWithManager (row):
    return 1 if row['YearsWithCurrManager'] >= 78.800082 else 0
```

In [47]:
```
df['YearsWithManager'] = df.apply(YearsWithManager,axis=1)
```

In [48]:
```
df['YearsWithManager'].value_counts()
```

Out[48]:
```
0    1470
Name: YearsWithManager, dtype: int64
```

In [49]:
```
numerical_col = ['Age','DailyRate','DistanceFromHome','Education','EmployeeCou
Scaled = StandardScaler()
Scaled= pd.DataFrame(Scaled.fit_transform(df[numerical_col]),columns=numerical
```

In [50]:
```
Scaled
```

Out[50]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	R
0	0.446350	0.742527	-1.010909	-0.891688	0.0	-1.701283	
1	1.322365	-1.297775	-0.147150	-1.868426	0.0	-1.699621	
2	0.008343	1.414363	-0.887515	-0.891688	0.0	-1.696298	
3	-0.429664	1.461466	-0.764121	1.061787	0.0	-1.694636	
4	-1.086676	-0.524295	-0.887515	-1.868426	0.0	-1.691313	
...	...	...	...	...	...	...	
1465	-0.101159	0.202082	1.703764	-0.891688	0.0	1.721670	
1466	0.227347	-0.469754	-0.393938	-1.868426	0.0	1.723332	
1467	-1.086676	-1.605183	-0.640727	0.085049	0.0	1.726655	
1468	1.322365	0.546677	-0.887515	0.085049	0.0	1.728317	
1469	-0.320163	-0.432568	-0.147150	0.085049	0.0	1.733302	

1470 rows × 16 columns

In [51]:
```
df
```

Out[51]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	E
0	41	Yes	Travel_Rarely	1102	Sales	1	2	
1	49	No	Travel_Frequently	279	Research & Development	8	1	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	
4	27	No	Travel_Rarely	591	Research & Development	2	1	
...	...	...	...	...	...	...	...	
1465	36	No	Travel_Frequently	884	Research & Development	23	2	
1466	39	No	Travel_Rarely	613	Research & Development	6	1	
1467	27	No	Travel_Rarely	155	Research & Development	4	3	
1468	49	No	Travel_Frequently	1023	Sales	2	3	
1469	34	No	Travel_Rarely	628	Research & Development	8	3	

1470 rows × 36 columns

In [52]:
```
x = pd.concat([encoder_cols,Scaled],axis=1)
y = df['StockOptionLevel']
```

In [53]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=
```

In [54]:
```
from sklearn.linear_model import LogisticRegression
```

In [55]:
```
log = LogisticRegression()
```

In [56]:
```
log.fit(x_train,y_train)
```

Out[56]:
```
LogisticRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [57]:
```
print("TrainScore",log.score(x_train,y_train))
```

```
TrainScore 1.0
```

In [58]:
```
print("Test Score",log.score(x_test,y_test))
```

```
Test Score 1.0
```

In [59]:
```
pred_train = log.predict(x_train)
pred_test = log.predict(x_test)
```

In [60]:
```
from sklearn import metrics
print(metrics.classification_report(y_train,pred_train))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       493
           1       1.00      1.00      1.00       483
           2       1.00      1.00      1.00       128
           3       1.00      1.00      1.00        72

    accuracy                           1.00      1176
   macro avg       1.00      1.00      1.00      1176
weighted avg       1.00      1.00      1.00      1176
```

In [65]:
```
from sklearn.metrics import matthews_corrcoef

mcc = matthews_corrcoef(y_test, pred_test)
print('MCC: ',mcc)
```

```
MCC:  1.0
```

In [66]:
```
from sklearn.model_selection import GridSearchCV
```

In [67]:
```
from sklearn.model_selection import GridSearchCV
param_grid ={
'penalty':["l1","l2"],
'C': [0.1,0.5,1,5,10]
}
grid=GridSearchCV(estimator=log, param_grid=param_grid, cv=5)
grid.fit(x_train,y_train)
```

Out[67]:
```
GridSearchCV(cv=5, estimator=LogisticRegression(),
             param_grid={'C': [0.1, 0.5, 1, 5, 10], 'penalty': ['l1', 'l2']})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [68]:
```
best_param = grid.best_params_
best_model = grid.best_estimator_
```

In [69]:
```
y_pred=best_model.predict(x_test)
```

In [70]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_
```

In [72]:
```
from sklearn.linear_model import Lasso, Ridge

lasso = Lasso(alpha=0.1)
lasso.fit(x_train, y_train)

ridge = Ridge(alpha=0.1)
ridge.fit(x_train, y_train)
```

Out[72]:
```
Ridge(alpha=0.1)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [73]:
```
from sklearn.model_selection import cross_val_score

scores = cross_val_score(model, x_train, y_train, cv=5)
print("Cross-validated accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.s
```

```
Cross-validated accuracy: 1.00 (+/- 0.00)
```

PERFORMING SVM

In [74]:
```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
from sklearn.metrics import confusion_matrix
```

In [75]:
```
scaler = StandardScaler()
```

In [76]:
```
x_train = scaler.fit_transform(x_train)
x_test = scaler.fit_transform(x_test)
```

In [77]:
```
svcm = SVC(kernel='linear')
```

In [78]:
```
svcm.fit(x_train,y_train)
```

Out[78]:
```
SVC(kernel='linear')
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [79]:
```
y_pred = svcm.predict(x_test)
```

In [80]:
```
y_pred
```

Out[80]:
```
array([0, 2, 1, 1, 0, 2, 0, 3, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 2, 0, 3,
       1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 3, 1, 1, 1, 0, 1, 0, 0, 1, 1,
       0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 3, 0, 0, 2, 0, 0, 0, 0, 0,
       1, 1, 0, 0, 2, 0, 1, 0, 2, 3, 1, 1, 1, 2, 1, 1, 1, 0, 1, 0, 0, 0,
       1, 0, 1, 0, 0, 1, 0, 0, 2, 2, 1, 1, 0, 3, 2, 1, 0, 2, 0, 0, 0, 1,
       2, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 3, 0, 0, 0, 3, 0, 1, 0, 0,
       0, 0, 1, 2, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0,
       2, 1, 1, 1, 2, 1, 2, 0, 2, 1, 1, 0, 0, 3, 1, 0, 2, 1, 1, 0, 0, 1,
       1, 1, 1, 0, 1, 0, 0, 0, 2, 3, 0, 0, 1, 0, 1, 0, 0, 1, 2, 0, 0, 1,
       1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 3, 0, 0, 0, 0, 3, 0, 2,
       1, 1, 2, 0, 1, 2, 2, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 2,
       0, 2, 1, 0, 1, 1, 1, 0, 0, 3, 0, 1, 1, 2, 0, 0, 1, 1, 2, 1, 2, 1,
       0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 2, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1,
       0, 0, 1, 0, 1, 0, 0, 1], dtype=int64)
```

In [81]:
```
acc = accuracy_score(y_test,y_pred)
print('Accuracy:{:.2f}%'. format(acc*100))
```

```
Accuracy:100.00%
```

In [82]:
```
print(classification_report(y_test,y_pred))
```

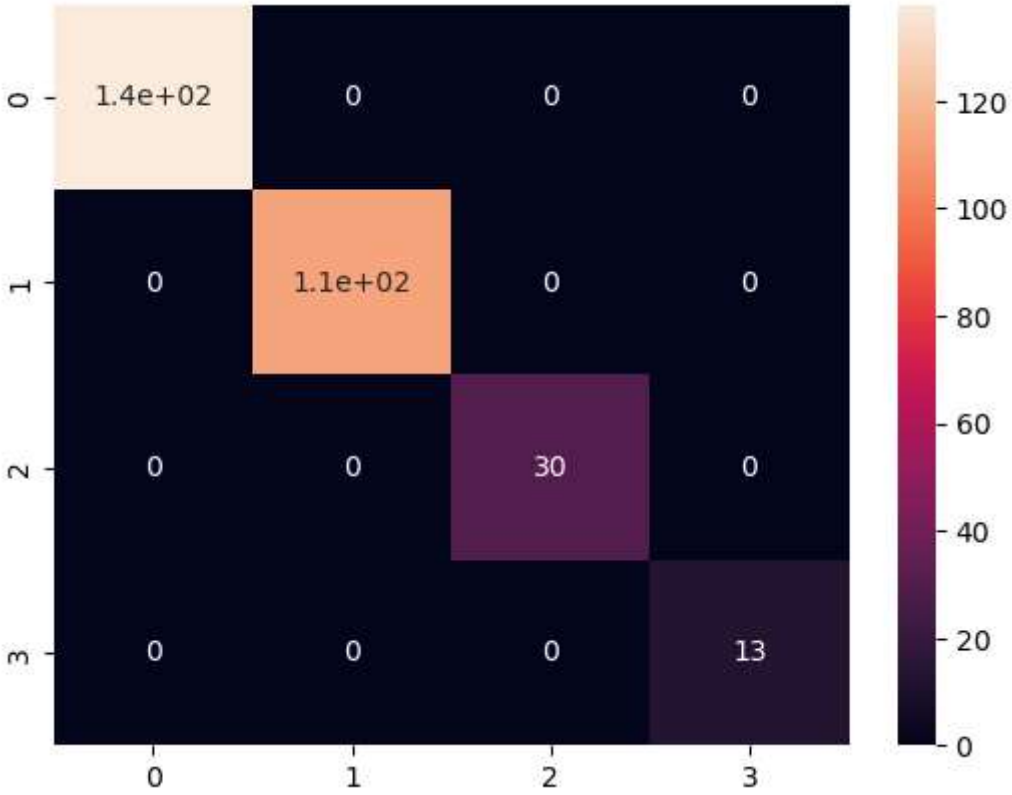
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       138
           1       1.00      1.00      1.00       113
           2       1.00      1.00      1.00        30
           3       1.00      1.00      1.00        13

    accuracy                           1.00       294
   macro avg       1.00      1.00      1.00       294
weighted avg       1.00      1.00      1.00       294
```

In [83]:
```
cm = confusion_matrix(y_test,y_pred)
print('confusion matrix:')
print(cm)
```

```
confusion matrix:
[[138   0   0   0]
 [  0 113   0   0]
 [  0   0  30   0]
 [  0   0   0  13]]
```

In [84]:
```
sns.heatmap(cm,annot=True)
```

Out[84]: <Axes: >



In [85]:
```
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import label_binarize
from sklearn.multiclass import OneVsRestClassifier
```

In [86]:
```
yb = label_binarize(y,classes=[0,1])
```

In [87]:
```
nc = yb.shape[1]
```

In [88]:
```
classifier = OneVsRestClassifier(SVC(kernel='linear',probability=True,random_s
```

In [89]:
```
y_score = classifier.fit(x_train,y_train).decision_function(x_test)
```

In [90]:
```
y_score_2d = y_score.reshape(-1, 1)
```

In [92]:
```
from sklearn.metrics import confusion_matrix
```

In [93]:
```
cm = confusion_matrix(y_test,y_pred)
print('confusion matrix:')
print(cm)
```

```
confusion matrix:
[[138   0   0   0]
 [  0 113   0   0]
 [  0   0  30   0]
 [  0   0   0  13]]
```

TUNING WITH GRID SEARCH CV

In [94]:
```
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
param_grid = {
 'C': [0.1, 1, 10, 100],
 'kernel': ['linear'],
}
svm = SVC()
grid_search = GridSearchCV(svm, param_grid, cv=5, n_jobs=-1)
grid_search.fit(x_train, y_train)
print("Best hyperparameters found: ", grid_search.best_params_)
print("Best accuracy on the validation set: {:.2f}".format(grid_search.best_sc
```

```
Best hyperparameters found:  {'C': 0.1, 'kernel': 'linear'}
Best accuracy on the validation set: 1.00
```

In [95]:
```
grid_search = GridSearchCV(svm, param_grid, cv=5, n_jobs=-1)
grid_search.fit(x_train, y_train)
```

Out[95]:
```
GridSearchCV(cv=5, estimator=SVC(), n_jobs=-1,
             param_grid={'C': [0.1, 1, 10, 100], 'kernel': ['linear']})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [96]:
```
print("Best hyperparameters found: ", grid_search.best_params_)
print("Best accuracy on the validation set: {:.2f}".format(grid_search.best_sc
```

```
Best hyperparameters found:  {'C': 0.1, 'kernel': 'linear'}
Best accuracy on the validation set: 1.00
```

In [97]:
```
best_svm = grid_search.best_estimator_
best_svm.fit(x_train, y_train)
```

Out[97]:
```
SVC(C=0.1, kernel='linear')
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [98]:
```
test_accuracy = best_svm.score(x_test, y_test)
print("Test accuracy: {:.2f}".format(test_accuracy))
```

```
Test accuracy: 1.00
```

In [99]:
```
test_accuracy = best_svm.score(x_train, y_train)
print("Test accuracy: {:.2f}".format(test_accuracy))
```

```
Test accuracy: 1.00
```

PERFORMING NAIVE BAYES

In [100]:
```
from sklearn import model_selection, naive_bayes, svm, metrics,feature_extrac
```

In [101]:
```
x = pd.concat([encoder_cols,Scaled],axis=1)
y = df['StockOptionLevel']
```

In [102]:
```
x_train,x_test,y_train,y_test= train_test_split(x,y,test_size=0.2,random_state
```

In [103]:
```
x_train
```

Out[103]:

	Attrition_Yes	BusinessTravel_Travel_Frequently	BusinessTravel_Travel_Rarely	Department_ & Dev
1097	0.0	0.0	1.0	
727	0.0	0.0	0.0	
254	0.0	0.0	1.0	
1175	0.0	0.0	1.0	
1341	0.0	0.0	1.0	
...	...	...	...	
1130	0.0	0.0	1.0	
1294	0.0	0.0	1.0	
860	1.0	1.0	0.0	
1459	0.0	0.0	1.0	
1126	0.0	0.0	1.0	

1176 rows × 38 columns

In [104]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

In [105]:
```
bayes = naive_bayes.MultinomialNB()
```

In [106]:
```
bayes.fit(x_train,y_train)
```

Out[106]:
```
MultinomialNB()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [107]:
```
y_pred_nb=bayes.predict(x_test)
```

In [108]:
```
accuracy=metrics.accuracy_score(y_test,y_pred_nb)
accuracy
```

Out[108]:
```
0.7687074829931972
```

In [109]:
```
print(metrics.classification_report(y_test, y_pred_nb))
```

```
              precision    recall  f1-score   support

           0       1.00      0.82      0.90       138
           1       0.62      1.00      0.77       113
           2       0.00      0.00      0.00        30
           3       0.00      0.00      0.00        13

    accuracy                           0.77       294
   macro avg       0.41      0.45      0.42       294
weighted avg       0.71      0.77      0.72       294
```

In [110]:
```
cm=confusion_matrix(y_test,y_pred)
cm
```

Out[110]:
```
array([[138,   0,   0,   0],
       [  0, 113,   0,   0],
       [  0,   0,  30,   0],
       [  0,   0,   0,  13]], dtype=int64)
```

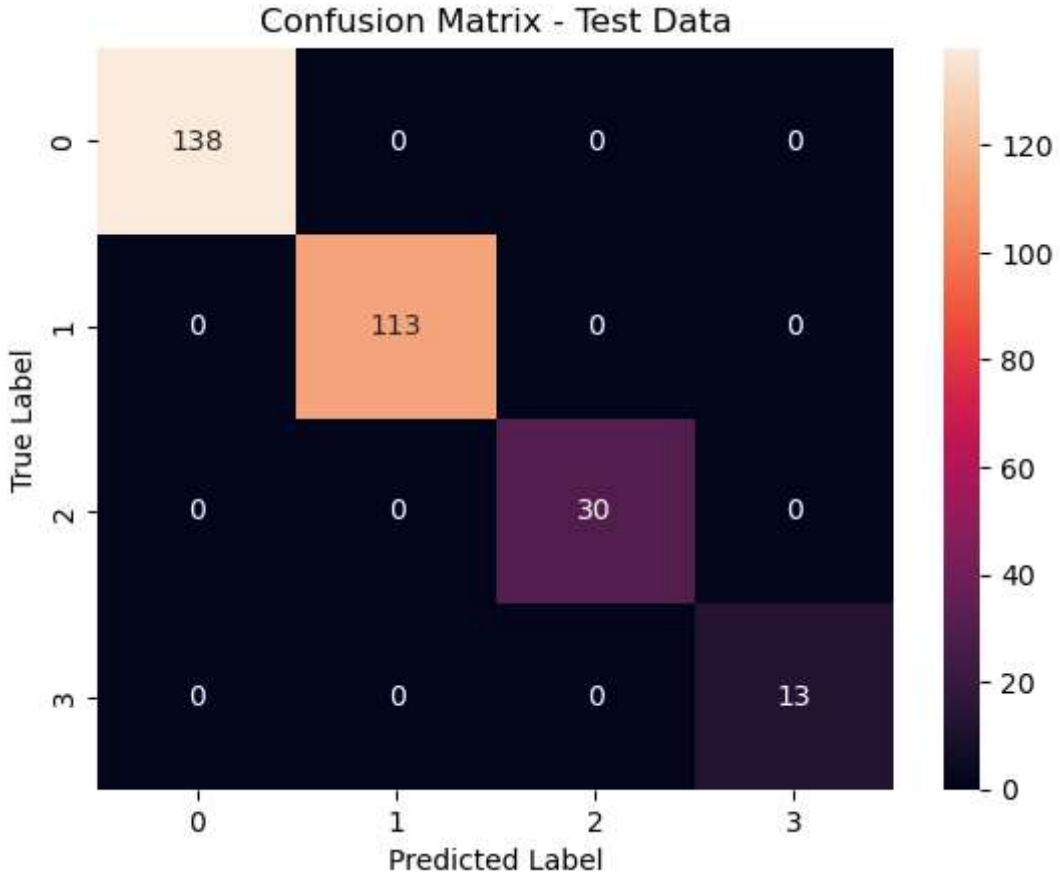
In [111]:
```
yb=label_binarize(y, classes=[0,1,2])
nc = yb.shape[1]
```

In [112]:
```
classifier = OneVsRestClassifier(bayes)
```

In [113]:
```
y_score=classifier.fit(x_train,y_train).predict(x_test)
```

In [116]:
```
cm=confusion_matrix(y_test,y_pred)
print(cm)
sns.heatmap(cm, annot=True, fmt='.3g')
plt.title('Confusion Matrix - Test Data')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

```
[[138   0   0   0]
 [  0 113   0   0]
 [  0   0  30   0]
 [  0   0   0  13]]
```



TUNING IN NAIVE BAYES

In [117]:
```
 param_grid = {
'alpha': [0.1, 1, 10, 100],
'fit_prior': [True, False]
}
```

In [118]:
```
bayes = naive_bayes.MultinomialNB()
grid_search = GridSearchCV(bayes, param_grid, cv=5)
grid_search.fit(x_train, y_train)
```

Out[118]:
```
GridSearchCV(cv=5, estimator=MultinomialNB(),
             param_grid={'alpha': [0.1, 1, 10, 100],
                         'fit_prior': [True, False]})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [119]:
```
best_param = grid_search.best_params_
best_nb = naive_bayes.MultinomialNB(alpha = best_param['alpha'], fit_prior = b
best_nb.fit(x_train, y_train)
y_pred_1 = best_nb.predict(x_test)
```

In [120]:
```
print("Best Hyperparameter : ", best_param)
```

```
Best Hyperparameter :  {'alpha': 0.1, 'fit_prior': True}
```

In [121]:
```
acc = accuracy_score(y_test, y_pred_1)
acc
```

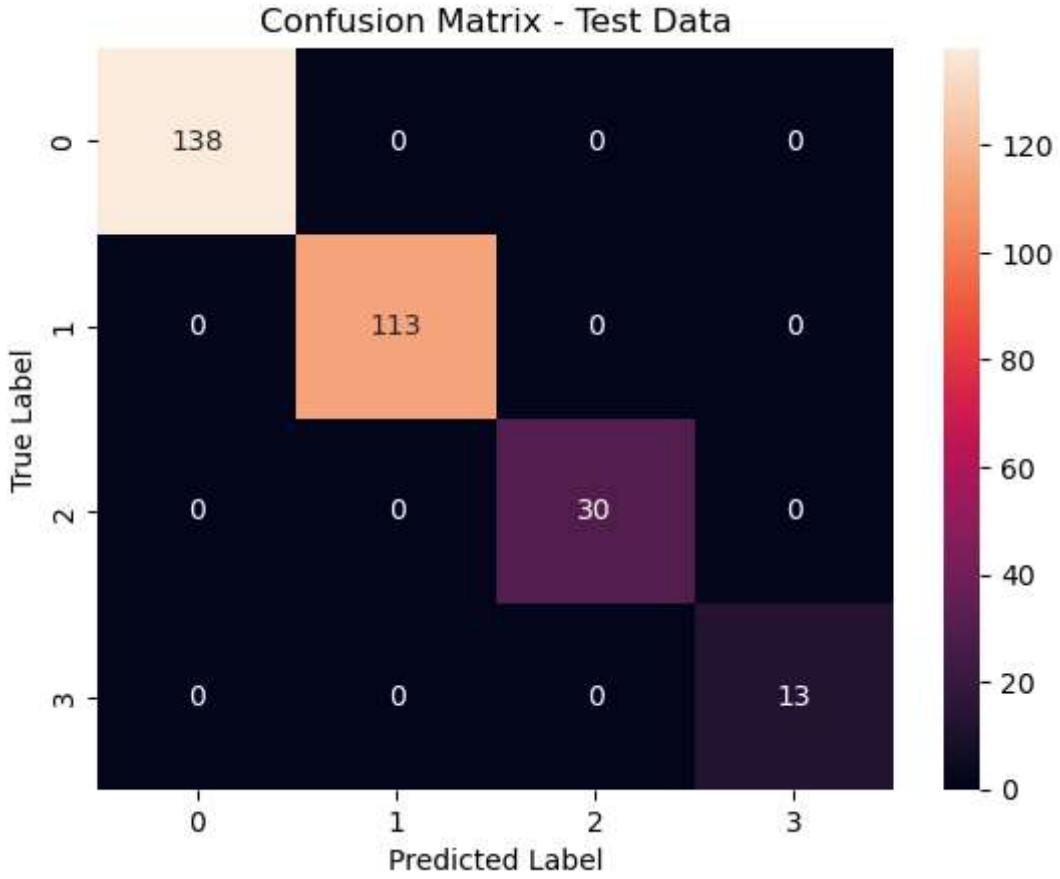
Out[121]:
```
0.7687074829931972
```

In [122]:
```
print (classification_report(y_test,y_pred_1))
```

```
              precision    recall  f1-score   support

           0       1.00      0.82      0.90       138
           1       0.62      1.00      0.77       113
           2       0.00      0.00      0.00        30
           3       0.00      0.00      0.00        13

    accuracy                           0.77       294
   macro avg       0.41      0.45      0.42       294
weighted avg       0.71      0.77      0.72       294
```

In [123]:
```
cm=confusion_matrix(y_test,y_pred)
print(cm)
sns.heatmap(cm, annot=True, fmt='.3g')
plt.title('Confusion Matrix - Test Data')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

```
[[138   0   0   0]
 [  0 113   0   0]
 [  0   0  30   0]
 [  0   0   0  13]]
```



In [124]:
```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import uniform
```

In [125]:
```
svm=SVC(kernel='linear')
param_dist={
    'C':uniform(loc=0,scale=10),
    'gamma':['scale','auto']+list(uniform(loc=0,scale=1).rvs(10)),
}
```

In [126]:
```
n_iter_search=20
random_search = RandomizedSearchCV(svcm, param_distributions=param_dist, n_ite
random_search.fit(x_train,y_train)
```

Out[126]:
```
RandomizedSearchCV(cv=5, estimator=SVC(kernel='linear'), n_iter=20, n_jobs=-
1,
                   param_distributions={'C': <scipy.stats._distn_infrastructu
re.rv_continuous_frozen object at 0x00000199B89DF490>,
                                        'gamma': ['scale', 'auto',
                                                  0.9361641890683409,
                                                  0.19628102401003744,
                                                  0.5284108463087878,
                                                  0.8132920126128795,
                                                  0.1381782095844173,
                                                  0.45161984754437445,
                                                  0.9180837631471861,
                                                  0.8587795855275844,
                                                  0.3758360165992314,
                                                  0.9279744367311602]},
                   random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [127]:
```
best_param = random_search.best_params_
best_model = random_search.best_estimator_
y_pred_2=best_model.predict(x_test)
```

In [128]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       138
           1       1.00      1.00      1.00       113
           2       1.00      1.00      1.00        30
           3       1.00      1.00      1.00        13

    accuracy                           1.00       294
   macro avg       1.00      1.00      1.00       294
weighted avg       1.00      1.00      1.00       294
```

In [129]:
```
cm= confusion_matrix(y_test,y_pred_2)
print(cm)
```

```
[[138   0   0   0]
 [  0 113   0   0]
 [  0   0  30   0]
 [  0   0   0  13]]
```