In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
data = pd.read_csv('Social_Network_Ads (2)-Copy1.csv')
```

In [3]:
```
data.head()
```

Out[3]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

In [4]:
```
data.tail()
```

Out[4]:

	User ID	Gender	Age	EstimatedSalary	Purchased
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   User ID          400 non-null    int64 
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64 
 3   EstimatedSalary  400 non-null    int64 
 4   Purchased        400 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

In [6]:
```
data.describe()
```

Out[6]:

	User ID	Age	EstimatedSalary	Purchased
count	4.000000e+02	400.000000	400.000000	400.000000
mean	1.569154e+07	37.655000	69742.500000	0.357500
std	7.165832e+04	10.482877	34096.960282	0.479864
min	1.556669e+07	18.000000	15000.000000	0.000000
25%	1.562676e+07	29.750000	43000.000000	0.000000
50%	1.569434e+07	37.000000	70000.000000	0.000000
75%	1.575036e+07	46.000000	88000.000000	1.000000
max	1.581524e+07	60.000000	150000.000000	1.000000

In [7]:
```
## preprocessing
```

In [8]:
```
data.isnull().sum()
```

Out[8]:
```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

In [9]:
```
data.nunique()
```

Out[9]:
```
User ID            400
Gender               2
Age                 43
EstimatedSalary    117
Purchased            2
dtype: int64
```

In [10]:
```
data.columns
```

Out[10]:
```
Index(['User ID', 'Gender', 'Age', 'EstimatedSalary', 'Purchased'], dtype='object')
```

In [11]:
```
df = pd.get_dummies(data['Gender'],drop_first=True)
```

In [12]:
```
data.drop(columns=['Gender'],inplace=True)
```

In [13]:
```
dataframe = pd.concat([df,data],axis=1)
```

In [14]:
```
dataframe.head()
```

Out[14]:

	Male	User ID	Age	EstimatedSalary	Purchased
0	1	15624510	19	19000	0
1	1	15810944	35	20000	0
2	0	15668575	26	43000	0
3	0	15603246	27	57000	0
4	1	15804002	19	76000	0

In [15]:
```
dataframe.drop(columns=['User ID'],inplace=True)
```

In [16]:
```
x = dataframe.drop(columns=['Purchased'])
y = dataframe['Purchased']
```

In [17]:
```
x.shape,y.shape
```

Out[17]:
```
((400, 3), (400,))
```

In [18]:
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import mean_squared_error,confusion_matrix,classification_report,accuracy_score
```

In [19]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,train_size=0.80,stratify=y,random_state=0)
```

In [20]:
```
x_train.shape,y_train.shape
```

Out[20]:
```
((320, 3), (320,))
```

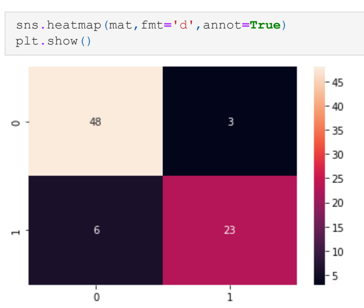
In [21]:
```
model = DecisionTreeClassifier()
model.fit(x_train,y_train)
y_pred= model.predict(x_test)
print('Accuracy :',accuracy_score(y_test,y_pred))
```

```
Accuracy : 0.8875
```

In [22]:
```
mat = confusion_matrix(y_test,y_pred)
mat
```

Out[22]:
```
array([[48,  3],
       [ 6, 23]], dtype=int64)
```

In [23]:
```
sns.heatmap(mat,fmt='d',annot=True)
plt.show()
```



In [24]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.89      0.94      0.91        51
           1       0.88      0.79      0.84        29

    accuracy                           0.89        80
   macro avg       0.89      0.87      0.88        80
weighted avg       0.89      0.89      0.89        80
```

In [25]:
```
model.predict([[1,30,30000]])
```

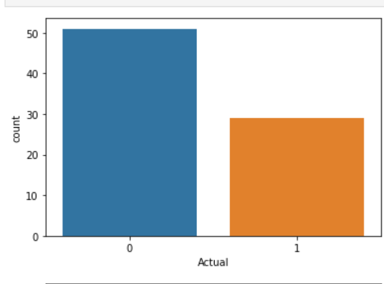
Out[25]:
```
array([0], dtype=int64)
```

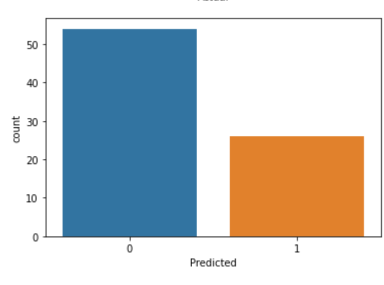
In [26]:
```
data1 = pd.DataFrame({'Actual':y_test,'Predicted':y_pred})
data1.head()
```

Out[26]:

	Actual	Predicted
361	1	1
245	1	1
313	1	0
294	0	0
376	0	0

In [27]:
```
sns.countplot(data1['Actual'])
plt.show()
sns.countplot(data1['Predicted'])
plt.show()
```





In [28]:
```
x_train.shape,y_train.shape
```

Out[28]:
```
((320, 3), (320,))
```

In [29]:
```
y_train.value_counts()
```

Out[29]:
```
0    206
1    114
Name: Purchased, dtype: int64
```

In [30]:
```
print('Value of o in y_train before oversampling :-',y_train.value_counts()[0])
print('Value of 1 in y_train before oversampling :-',y_train.value_counts()[1])
```

```
Value of o in y_train before oversampling :- 206
Value of 1 in y_train before oversampling :- 114
```

In [31]:
```
import imblearn
```

In [32]:
```
from imblearn.over_sampling import SMOTE
```

In [33]:
```
sm = SMOTE(sampling_strategy=1,random_state=42,n_jobs=-1,k_neighbors=5)
```

In [34]:
```
x_train = np.array(x_train)
y_train = np.array(y_train)
```

In [36]:
```
x_train.shape,y_train.shape
```

Out[36]:
```
((320, 3), (320,))
```

In [37]:
```
x_train_res,y_train_res = sm.fit_resample(x_train,y_train.ravel())
```

In [38]:
```
model = DecisionTreeClassifier()
model.fit(x_train_res,y_train_res)
y_predi= model.predict(x_test)
print('Accuracy :',accuracy_score(y_test,y_predi))
```

```
Accuracy : 0.9125
```

In [39]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.89      0.94      0.91        51
           1       0.88      0.79      0.84        29

    accuracy                           0.89        80
   macro avg       0.89      0.87      0.88        80
weighted avg       0.89      0.89      0.89        80
```

In [40]:
```
print('Accuarcy of training data :-',model.score(x_train_res,y_train_res))
print('Accuarcy of testing data :-',model.score(x_test,y_test))
```

```
Accuarcy of training data :- 1.0
Accuarcy of testing data :- 0.9125
```

CRoss validation behaviour.

In [41]:
```
from sklearn.model_selection import cross_val_score
```

In [47]:
```
val_score = cross_val_score(model,x_train_res,y_train_res,cv=10)
val_score
```

Out[47]:
```
array([0.85714286, 0.95238095, 0.85365854, 0.92682927, 0.82926829,
       0.85365854, 0.90243902, 0.90243902, 0.90243902, 0.85365854])
```

In [49]:
```
import numpy as np
variance = np.var(val_score)
print('Variance :-',variance)
mean = np.mean(val_score)
print('mean :-  ',mean)
std = np.std(val_score)
print('Standard Deviation :- ',std)
```

```
Variance :- 0.0014005268972550364
mean :-   0.8833914053426248
Standard Deviation :-  0.03742361416612559
```

Model seems good for unseen data because the standard deviation is very less and all scores are getting close the mean.

In [50]:
```
val_score_mean = cross_val_score(model,x_train_res,y_train_res,cv=10).mean()*100
val_score_mean
```

Out[50]:
```
88.58304297328687
```

In [51]:
```
val_score_max = cross_val_score(model,x_train_res,y_train_res,cv=10).max()*100
val_score_max
```

Out[51]:
```
95.23809523809523
```

In [52]:
```
val_score_min = cross_val_score(model,x_train_res,y_train_res,cv=10).min()*100
val_score_min
```

Out[52]:
```
83.33333333333334
```

In [53]:
```
sqrt_error= np.sqrt(mean_squared_error(y_test,y_predi))
sqrt_error
```

Out[53]:
```
0.2958039891549808
```

In [54]:
```
np.std(y_test)
```

Out[54]:
```
0.4807221130757353
```

- HERE WE Can say the standard deviation of the actual value is greter than error occure due the classification.That means our model is good model.

AUC-ROC Curve.

In [56]:
```
from plotnine import aes,geom_area,geom_abline,geom_line,ggplot,ggtitle
```

In [57]:
```
from sklearn.metrics import auc,roc_curve
```

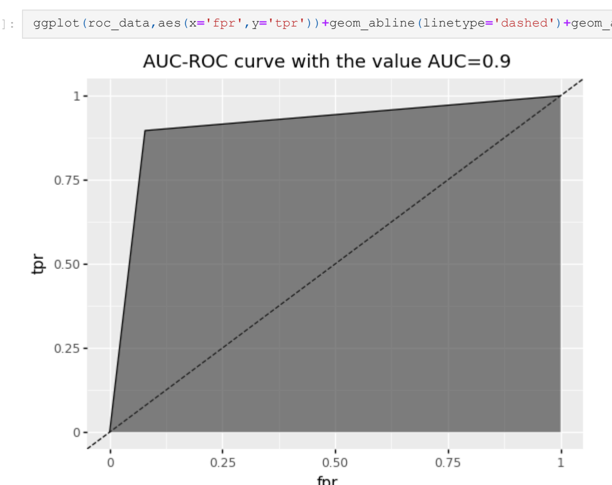
In [59]:
```
probab=model.predict_proba(x_test)[:,1]
```

In [60]:
```
fpr,tpr,thresh = roc_curve(y_test,probab)
```

In [61]:
```
roc_data = pd.DataFrame(dict(fpr=fpr,tpr=tpr))
```

In [62]:
```
auc= round(auc(fpr,tpr),1)
```

In [63]:
```
ggplot(roc_data,aes(x='fpr',y='tpr'))+geom_abline(linetype='dashed')+geom_area(alpha=0.6)+geom_line()+ggtitle('
```



Out[63]:
```
<ggplot: (97989171520)>
```

RandomForest Algorithm

In [64]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV
```

In [65]:
```
classifier = RandomForestClassifier()
```

In [66]:
```
from scipy.stats import randint
```

In [67]:
```
params = {'n_estimators':randint(100,400),'criterion':['gini','entropy'],'min_samples_leaf':randint(1,5)}
```

In [68]:
```
clf_cv = RandomizedSearchCV(classifier,params,n_iter=10,cv=10)
```

In [69]:
```
clf_cv.fit(x_train_res,y_train_res)
```

Out[69]:
```
RandomizedSearchCV(cv=10, estimator=RandomForestClassifier(),
                   param_distributions={'criterion': ['gini', 'entropy'],
                                        'min_samples_leaf': <scipy.stats._distn_infrastructure.rv_frozen object
at 0x0000016D09CA99D0>,
                                        'n_estimators': <scipy.stats._distn_infrastructure.rv_frozen object at
0x0000016D09C26940>})
```

In [70]:
```
clf_cv.best_params_
```

Out[70]:
```
{'criterion': 'entropy', 'min_samples_leaf': 4, 'n_estimators': 194}
```

In [71]:
```
clf_cv.best_score_
```

Out[71]:
```
0.9319396051103368
```

In [72]:
```
Y_predict=clf_cv.predict(x_test)
```

In [73]:
```
mat =confusion_matrix(y_test,Y_predict)
mat
```

Out[73]:
```
array([[46,  5],
       [ 3, 26]], dtype=int64)
```

In [74]:
```
sns.heatmap(mat,fmt='d',annot=True)
plt.show()
```

In [75]:
```
print(classification_report(y_test,Y_predict))
```

```
              precision    recall  f1-score   support

           0       0.94      0.90      0.92        51
           1       0.84      0.90      0.87        29

    accuracy                           0.90        80
   macro avg       0.89      0.90      0.89        80
weighted avg       0.90      0.90      0.90        80
```

In [76]:
```
sqrt_err=np.sqrt(mean_squared_error(y_test,Y_predict))
sqrt_err
```

Out[76]:
```
0.31622776601683794
```

In [77]:
```
np.std(y_test)
```

Out[77]:
```
0.4807221130757353
```

- Since we are getting the sqrt error lesser than the standard deviation of actual.That's means our model is good enough.

To get the more information regarding to regarding to the our model whether model overfitted or not.

In [78]:
```
print('Accuarcy of training data :-',clf_cv.score(x_train_res,y_train_res))
print('Accuarcy of testing data :-',clf_cv.score(x_test,y_test))
```

```
Accuarcy of training data :- 0.9368932038834952
Accuarcy of testing data :- 0.9
```

- Since model train on each and every data and predict outof which 94 % data that means our model good in training as well as if we see the model testing accuarcy got increasing upto 90%.Since it is lesser than the training data that's means our model is bit overfitted.

AUC_ROC Curve.

In [79]:
```
from sklearn.metrics import roc_curve,auc
from plotnine import aes,geom_abline,geom_area,geom_line,ggplot,ggtitle
```

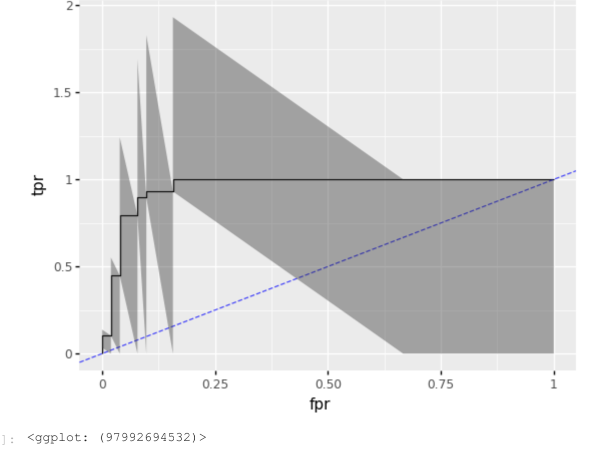
In [80]:
```
probablities=clf_cv.predict_proba(x_test)[:,1]
```

In [81]:
```
fpr,tpr,thresh=roc_curve(y_test,probablities)
```

In [82]:
```
roc_data = pd.DataFrame(dict(tpr=tpr,fpr=fpr))
```

In [83]:
```
auc = round(auc(fpr,tpr),1)
```

In [86]:
```
ggplot(roc_data,aes(x='fpr',y='tpr'))+geom_line()+geom_abline(linetype='dashed',color='blue',alpha=0.7)+geom_ar
```



Out[86]:
```
<ggplot: (97992694532)>
```

Thank You..!!