SUYASH PRATAP SINGH

Registration ID: GO_STP_6412 ¶

Tasks:-

Predict retention of an employee within an organization such that whether the employee will leave the company or continue with it. An organization is only as good as its employees, and these people are the true source of its competitive advantage. Dataset is downloaded from Kaggle. Link: https://www.kaggle.com/giripujar/hr-analytics (https://www.kaggle.com/giripujar/hr-analytics)

First do data exploration and visualization, after this create a logistic regression model to predict Employee Attrition Using Machine Learning & Python.

Import The Required Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from pingouin import qqplot
```

Reading The CSV file

In [2]:
```
kp = pd.read_csv(r'C:\Users\Admin\Desktop\Covid\HR_comma_sep.csv')
```

In [3]:
```
kp.head()  #print first five row
```

Out[3]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_compan
0	0.38	0.53	2	157	
1	0.80	0.86	5	262	
2	0.11	0.88	7	272	
3	0.72	0.87	5	223	
4	0.37	0.52	2	159	

In [4]:
```
kp.sample(4)  #print 4 random row
```

Out[4]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_con
9261	0.80	0.56	6	111	
8858	0.73	0.60	4	222	
10845	0.77	0.91	4	161	
4436	0.83	0.57	3	137	

Checking how many null values are there in dataset

In [5]:
```
kp.isna().sum()
```

Out[5]:
```
satisfaction_level       0
last_evaluation          0
number_project           0
average_montly_hours     0
time_spend_company       0
Work_accident            0
left                     0
promotion_last_5years    0
Department               0
salary                   0
dtype: int64
```

Renaming certain columns for better readability

In [6]:
```
kp = kp.rename(columns={'satisfaction_level': 'satisfaction',
                        'last_evaluation': 'evaluation',
                        'number_project': 'projectCount',
                        'average_montly_hours': 'averageMonthlyHours',
                        'time_spend_company': 'yearsAtCompany',
                        'Work_accident': 'workAccident',
                        'promotion_last_5years': 'promotion',
                        'sales' : 'department',
                        'left' : 'turnover'
                        })
```

Shape of dataset

In [7]:
```
kp.shape
```

Out[7]:
```
(14999, 10)
```

In [8]:
```
kp.dtypes
```

Out[8]:
```
satisfaction           float64
evaluation             float64
projectCount             int64
averageMonthlyHours      int64
yearsAtCompany           int64
workAccident             int64
turnover                 int64
promotion                int64
Department              object
salary                  object
dtype: object
```

In [9]:
```
kp.size
```

Out[9]:
```
149990
```

Descriptive analysis of the dataset using describe function

In [10]:
```
kp.describe()
```

Out[10]:

	satisfaction	evaluation	projectCount	averageMonthlyHours	yearsAtCompany	workA
count	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999
mean	0.612834	0.716102	3.803054	201.050337	3.498233	0
std	0.248631	0.171169	1.232592	49.943099	1.460136	0
min	0.090000	0.360000	2.000000	96.000000	2.000000	0
25%	0.440000	0.560000	3.000000	156.000000	3.000000	0
50%	0.640000	0.720000	4.000000	200.000000	3.000000	0
75%	0.820000	0.870000	5.000000	245.000000	4.000000	0
max	1.000000	1.000000	7.000000	310.000000	10.000000	1

In [11]:
```
print(kp.mean()) #mean
print(kp.std())  #standard deviation
```

```
satisfaction            0.612834
evaluation              0.716102
projectCount            3.803054
averageMonthlyHours    201.050337
yearsAtCompany          3.498233
workAccident            0.144610
turnover                0.238083
promotion               0.021268
dtype: float64
satisfaction            0.248631
evaluation              0.171169
projectCount            1.232592
averageMonthlyHours    49.943099
yearsAtCompany          1.460136
workAccident            0.351719
turnover                0.425924
promotion               0.144281
dtype: float64
```

Basic information about the dataset

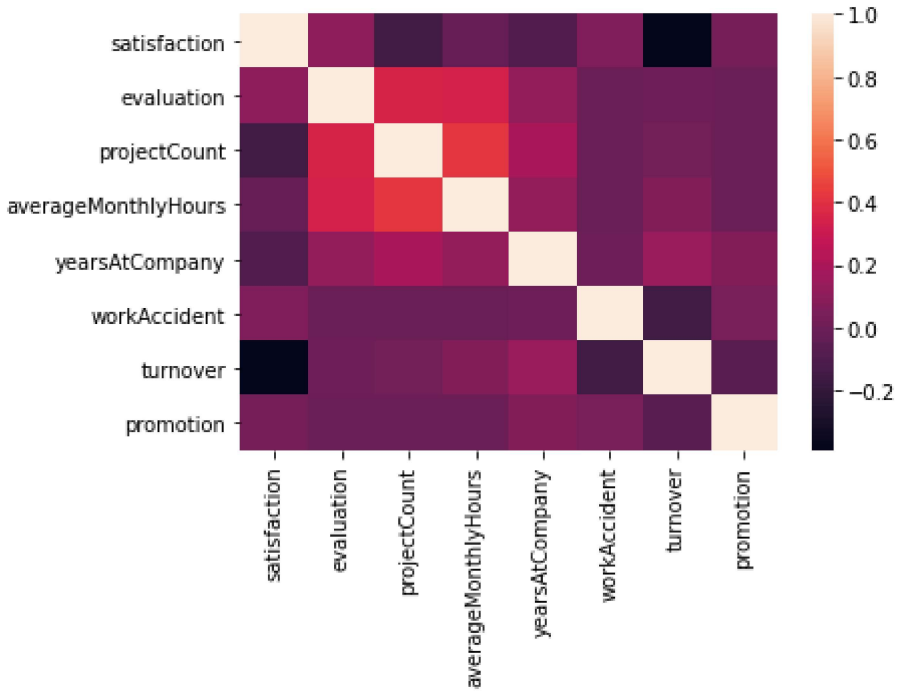
In [12]:
```
kp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 10 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   satisfaction         14999 non-null  float64
 1   evaluation           14999 non-null  float64
 2   projectCount         14999 non-null  int64
 3   averageMonthlyHours  14999 non-null  int64
 4   yearsAtCompany       14999 non-null  int64
 5   workAccident         14999 non-null  int64
 6   turnover             14999 non-null  int64
 7   promotion            14999 non-null  int64
 8   Department           14999 non-null  object
 9   salary               14999 non-null  object
dtypes: float64(2), int64(6), object(2)
memory usage: 1.1+ MB
```

In [13]:
```
#Correlation Matrix
corr = kp.corr()
corr = (corr)
sns.heatmap(corr,
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values)

corr
```

Out[13]:

	satisfaction	evaluation	projectCount	averageMonthlyHours	yearsAtCompa
satisfaction	1.000000	0.105021	-0.142970	-0.020048	-0.1008
evaluation	0.105021	1.000000	0.349333	0.339742	0.1315
projectCount	-0.142970	0.349333	1.000000	0.417211	0.1967
averageMonthlyHours	-0.020048	0.339742	0.417211	1.000000	0.1277
yearsAtCompany	-0.100866	0.131591	0.196786	0.127755	1.0000
workAccident	0.058697	-0.007104	-0.004741	-0.010143	0.0021
turnover	-0.388375	0.006567	0.023787	0.071287	0.1448
promotion	0.025605	-0.008684	-0.006064	-0.003544	0.0674



Compare the means of our employee turnover satisfaction against the employee population satisfaction

In [14]:
```
emp_population = kp['satisfaction'][kp['turnover'] == 0].mean()
emp_turnover_satisfaction = kp[kp['turnover']==1]['satisfaction'].mean()

print( 'The mean satisfaction for the employee population with no turnover is:
' + str(emp_population))
print( 'The mean satisfaction for employees that had a turnover is: ' + str(em
p_turnover_satisfaction) )
```

```
The mean satisfaction for the employee population with no turnover is: 0.6668
09590479524
The mean satisfaction for employees that had a turnover is: 0.440098011761411
4
```

Conducting the T-Test

In [15]:
```
import scipy.stats as stats
stats.ttest_1samp(a=  kp[kp['turnover']==1]['satisfaction'],popmean = emp_popu
lation)
```

Out[15]:
```
Ttest_1sampResult(statistic=-51.33034867547431, pvalue=0.0)
```

T-Test Quantile

In [16]:
```
degree_freedom = len(kp[kp['turnover']==1])

LQ = stats.t.ppf(0.025,degree_freedom)  # Left Quartile

RQ = stats.t.ppf(0.975,degree_freedom)  # Right Quartile

print ('The t-distribution left quartile range is: ' + str(LQ))
print ('The t-distribution right quartile range is: ' + str(RQ))
```

```
The t-distribution left quartile range is: -1.9606285215955626
The t-distribution right quartile range is: 1.9606285215955621
```

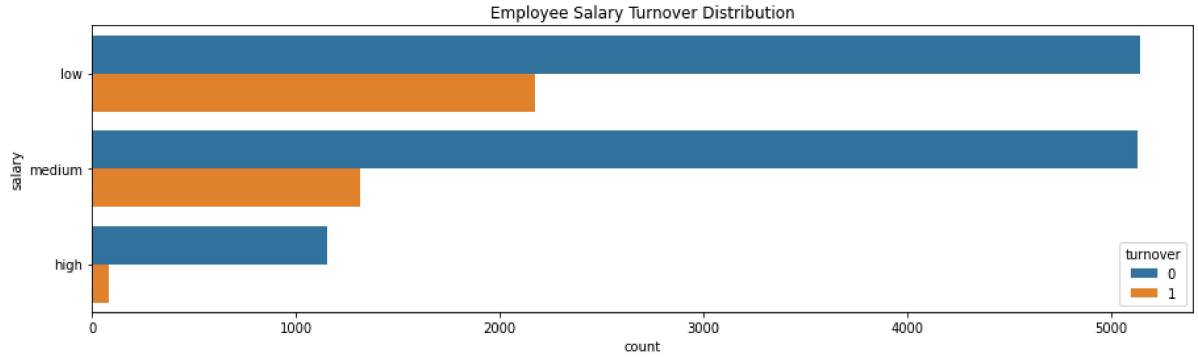
Conclusion:-

Reject the null hypothesis because:

T-Test score is outside the quantiles P-value is lower than confidence level of 5%

Data Visualization:-

In [17]:
```
f, ax = plt.subplots(figsize=(15, 4))
sns.countplot(y="salary", hue='turnover', data=kp).set_title('Employee Salary
 Turnover Distribution');
```



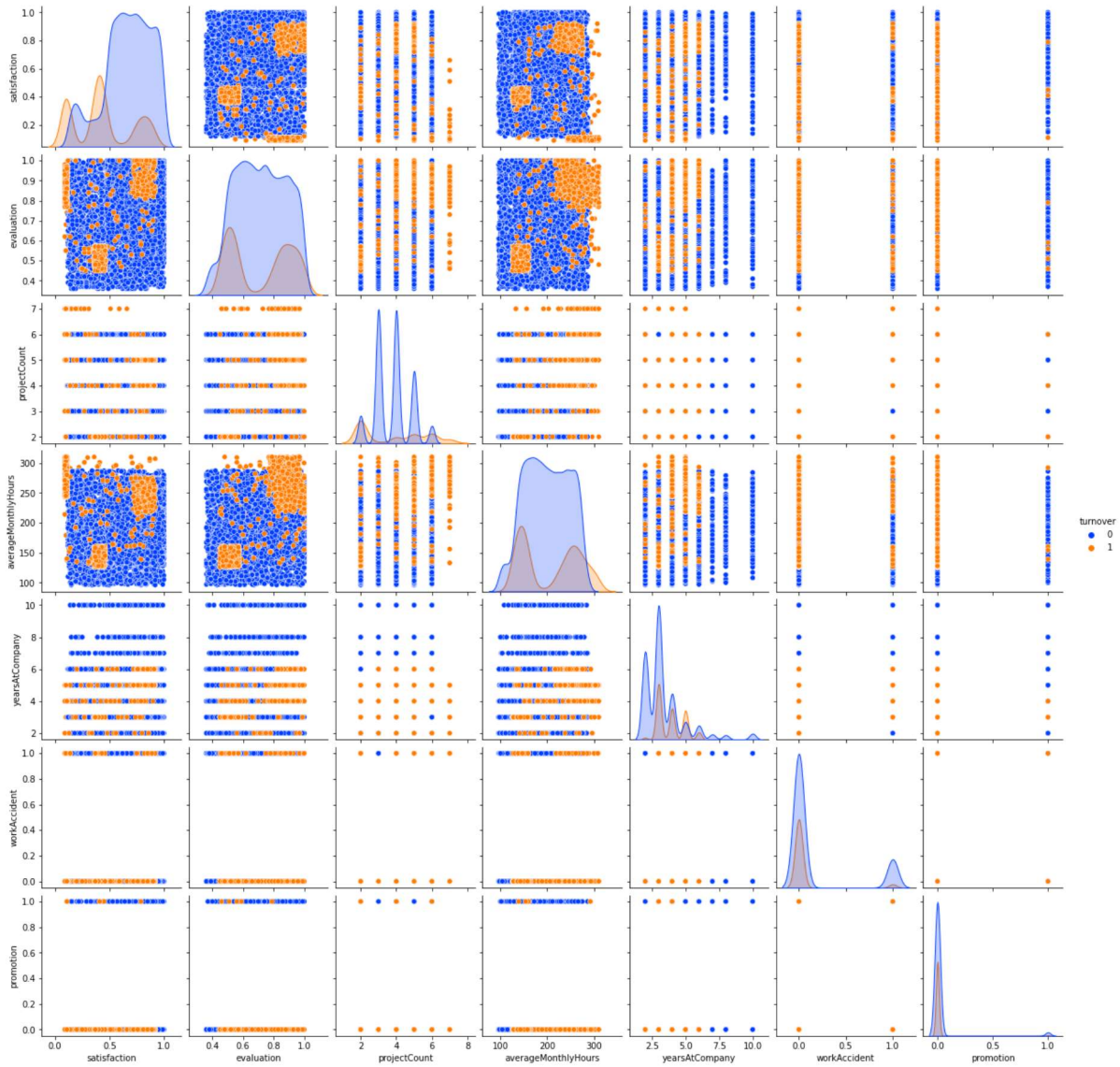
In [18]:
```
kp.head(1)
```

Out[18]:

	satisfaction	evaluation	projectCount	averageMonthlyHours	yearsAtCompany	workAccident
0	0.38	0.53	2	157	3	0

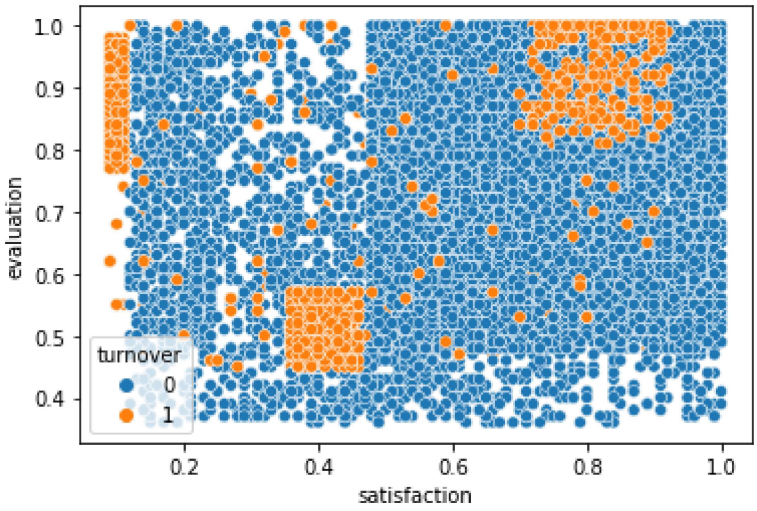
In [19]:
```
sns.pairplot(kp, hue="turnover",palette='bright')
```

Out[19]: <seaborn.axisgrid.PairGrid at 0x24fe64d5c40>



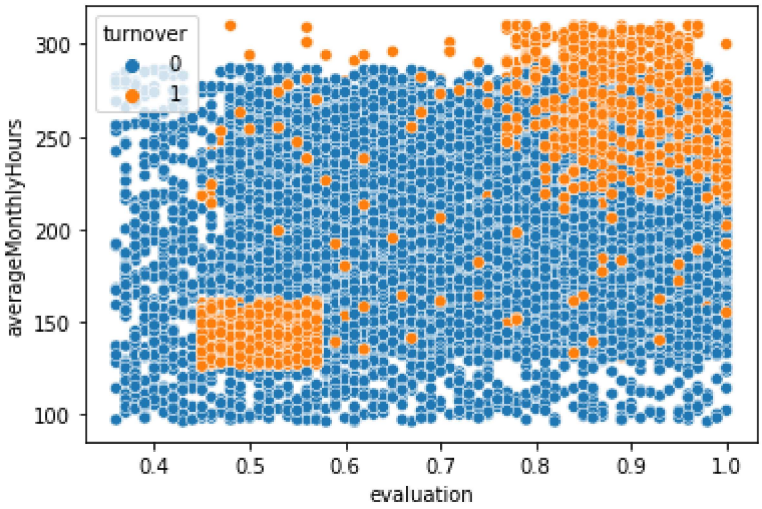
In [20]:
```
sns.scatterplot(data=kp,x='satisfaction',y='evaluation',hue='turnover')
```

Out[20]: <AxesSubplot:xlabel='satisfaction', ylabel='evaluation'>



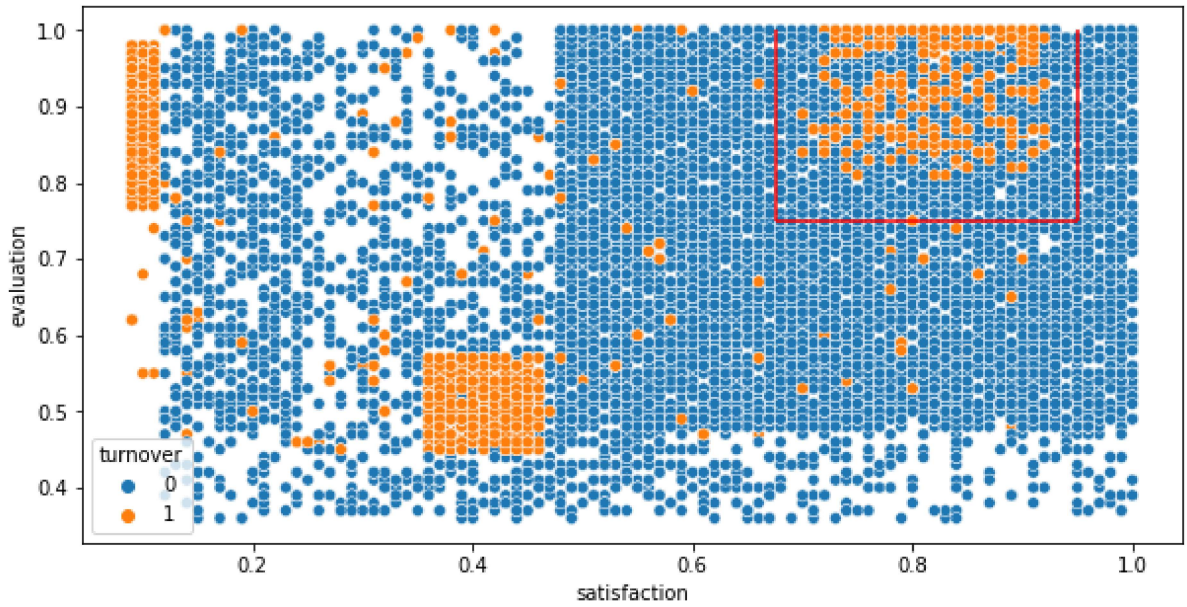
In [21]:
```
sns.scatterplot(data=kp,x='evaluation',y='averageMonthlyHours',hue='turnover')
```

Out[21]: <AxesSubplot:xlabel='evaluation', ylabel='averageMonthlyHours'>



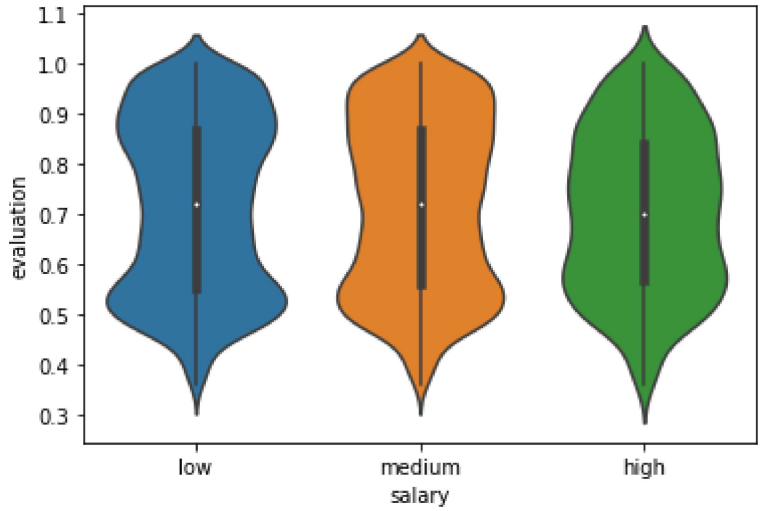
In [22]:
```
fig=plt.figure(figsize=(10,5))
sns.scatterplot(data=kp,x='satisfaction',y='evaluation',hue='turnover')
plt.vlines(0.675,0.75,1.0,'red')
plt.hlines(0.75,0.675,0.95,'red')
plt.vlines(0.95,0.75,1.0,'red')
```

Out[22]: <matplotlib.collections.LineCollection at 0x24fedff51c0>

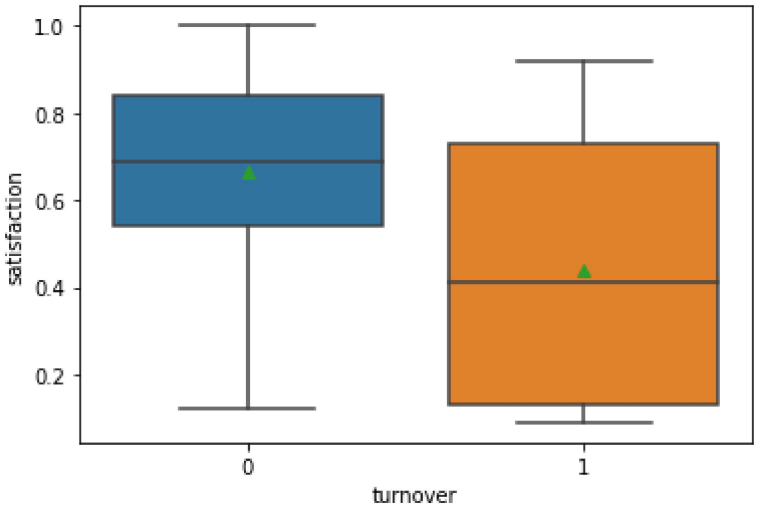


In [23]:
```
sns.violinplot(x=kp.salary,y=kp['evaluation'])
```

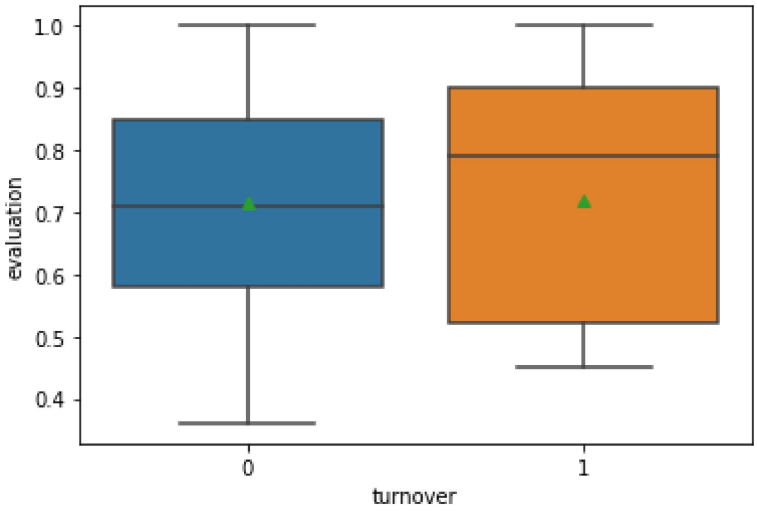
Out[23]: <AxesSubplot:xlabel='salary', ylabel='evaluation'>



In [24]:
```
box=sns.boxplot(data = kp,y='satisfaction',x='turnover',showmeans=True)
```



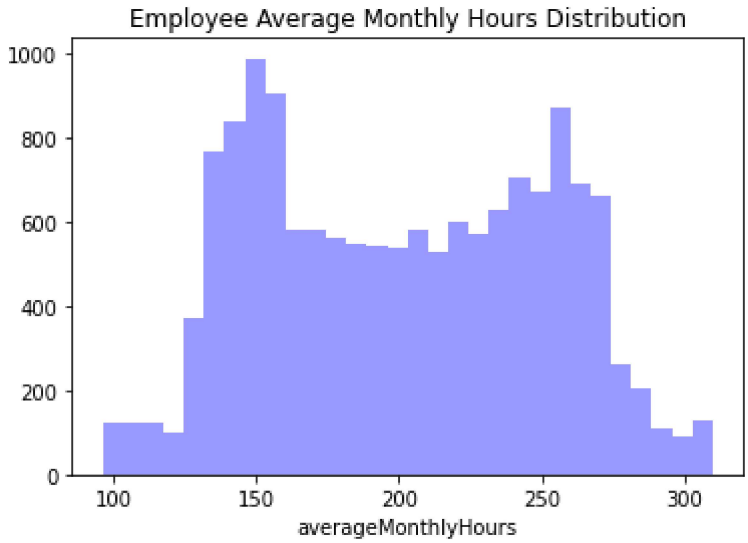
In [25]:
```
box=sns.boxplot(data = kp,y='evaluation',x='turnover',showmeans=True)
```



In [26]:
```
sns.distplot(kp.averageMonthlyHours, kde=False, color="b").set_title('Employee Average Monthly Hours Distribution')
```

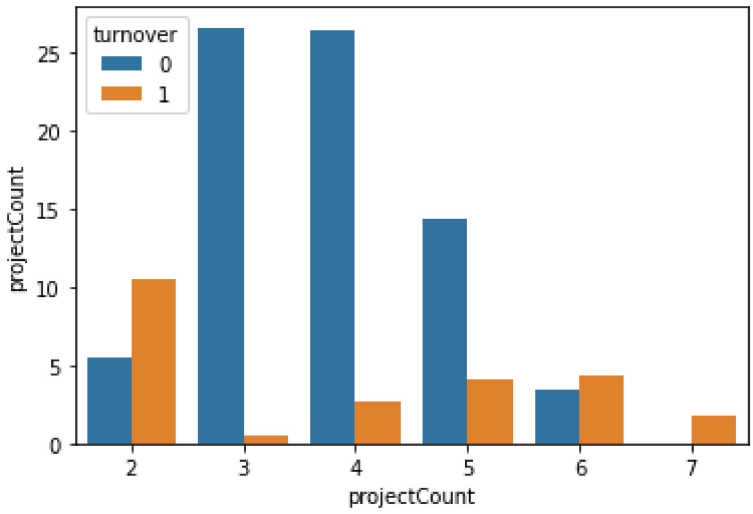
```
c:\users\admin\appdata\local\programs\python\python38\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[26]:
```
Text(0.5, 1.0, 'Employee Average Monthly Hours Distribution')
```



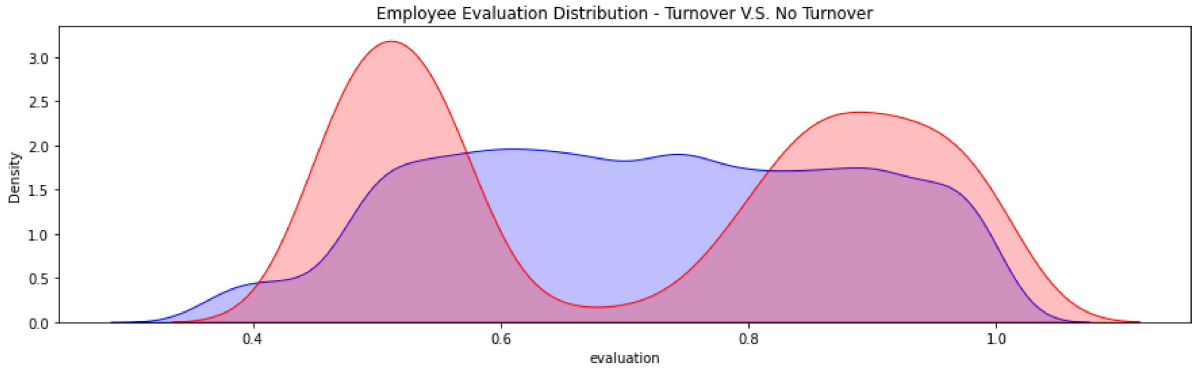
In [27]:
```
sns.barplot(x="projectCount", y="projectCount", hue="turnover", data=kp, estimator=lambda x: len(x) / len(kp) * 100)
```

Out[27]:
```
<AxesSubplot:xlabel='projectCount', ylabel='projectCount'>
```



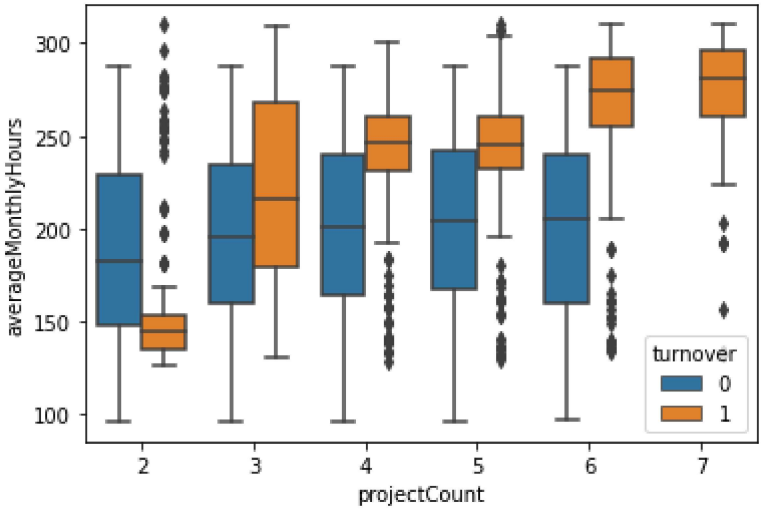
In [28]:
```
# Kernel Density Plot
fig = plt.figure(figsize=(15,4),)
ax=sns.kdeplot(kp.loc[(kp['turnover'] == 0),'evaluation'] , color='b',shade=True,label='no turnover')
ax=sns.kdeplot(kp.loc[(kp['turnover'] == 1),'evaluation'] , color='r',shade=True, label='turnover')
plt.title('Employee Evaluation Distribution - Turnover V.S. No Turnover')
```

Out[28]:
```
Text(0.5, 1.0, 'Employee Evaluation Distribution - Turnover V.S. No Turnover')
```



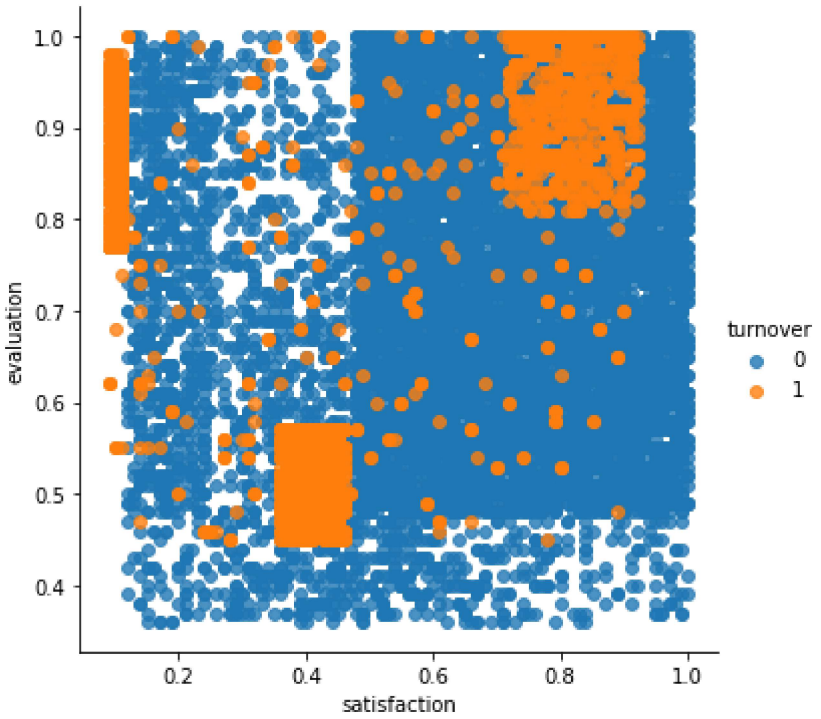
In [29]:
```
sns.boxplot(x="projectCount", y="averageMonthlyHours", hue="turnover", data=kp)
```

Out[29]:
```
<AxesSubplot:xlabel='projectCount', ylabel='averageMonthlyHours'>
```



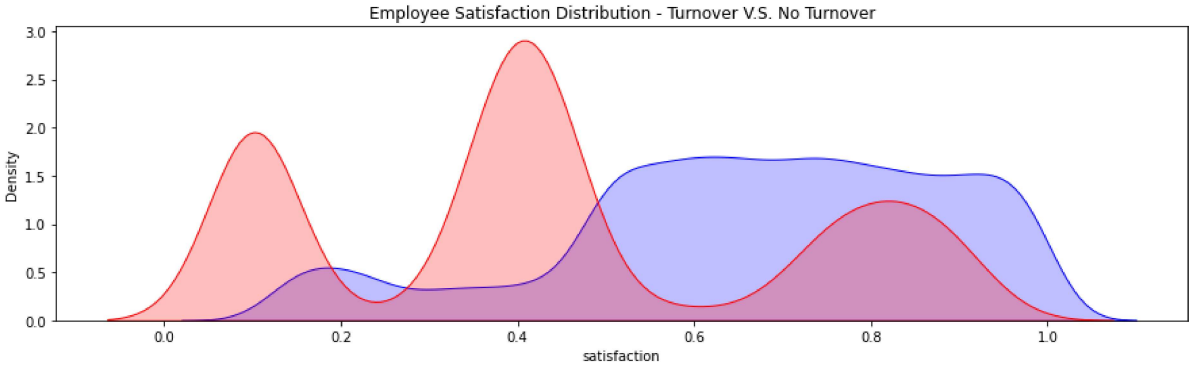
In [30]:
```
sns.lmplot(x='satisfaction', y='evaluation', data=kp, fit_reg=False, hue='turnover')
```

Out[30]:
```
<seaborn.axisgrid.FacetGrid at 0x24fee3f7310>
```



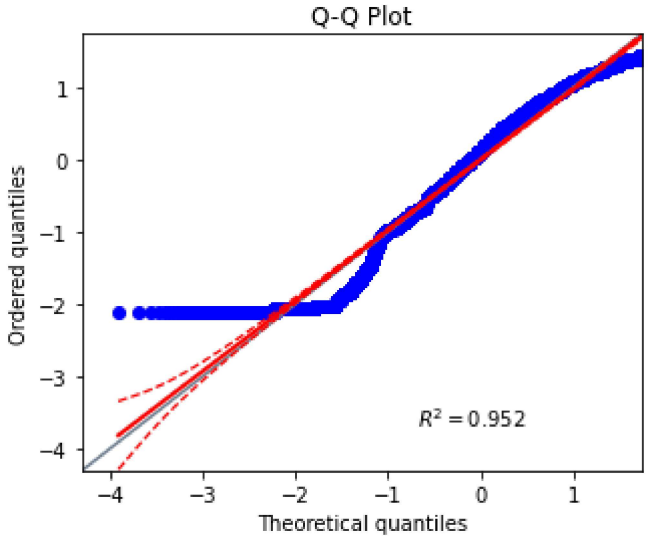
In [31]:
```
#KDEPlot: Kernel Density Estimate Plot
fig = plt.figure(figsize=(15,4))
ax=sns.kdeplot(kp.loc[(kp['turnover'] == 0),'satisfaction'] , color='b',shade=True, label='no turnover')
ax=sns.kdeplot(kp.loc[(kp['turnover'] == 1),'satisfaction'] , color='r',shade=True, label='turnover')
plt.title('Employee Satisfaction Distribution - Turnover V.S. No Turnover')
```

Out[31]:
```
Text(0.5, 1.0, 'Employee Satisfaction Distribution - Turnover V.S. No Turnover')
```



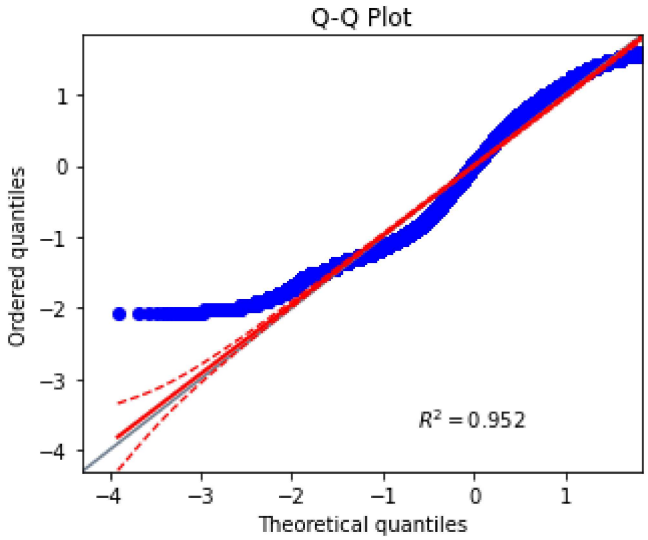
In [32]:
```
qqplot(kp['satisfaction'], dist='norm')
```

Out[32]:
```
<AxesSubplot:title={'center':'Q-Q Plot'}, xlabel='Theoretical quantiles', yla
bel='Ordered quantiles'>
```



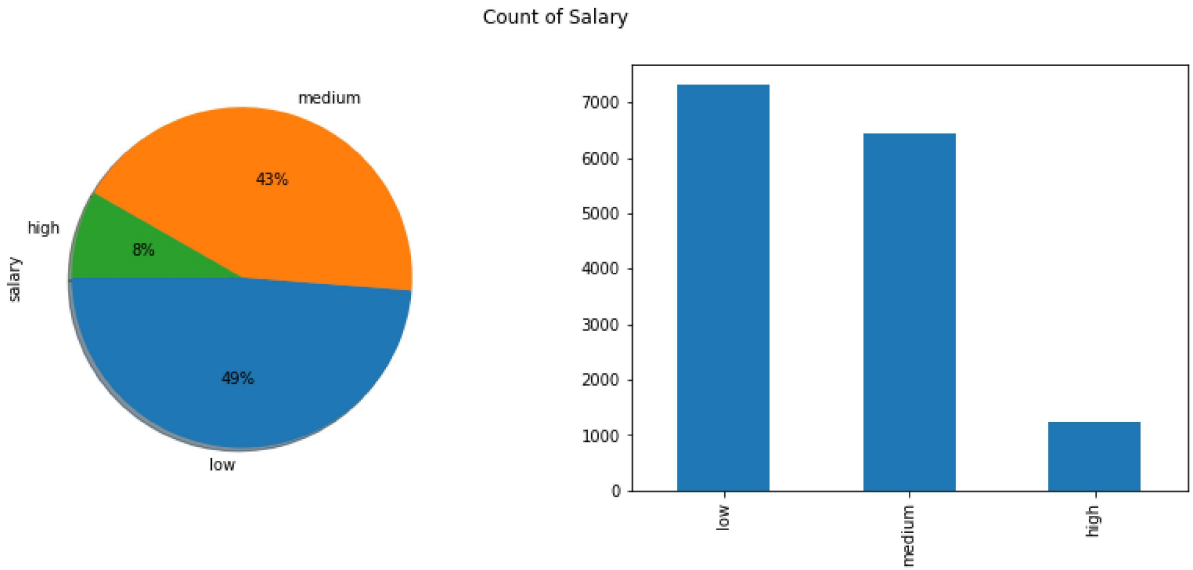
In [33]:
```
qqplot(kp['evaluation'], dist='norm')
```

Out[33]:
```
<AxesSubplot:title={'center':'Q-Q Plot'}, xlabel='Theoretical quantiles', yla
bel='Ordered quantiles'>
```



In [34]:
```
fig,ax=plt.subplots(ncols=2,figsize=(14,5))
kp.salary.value_counts().plot.pie(autopct='%.0f%%',labels=kp.salary.unique(),s
hadow = True,startangle = 180,ax=ax[0])
kp.salary.value_counts().plot.bar()
fig.suptitle('Count of Salary')
plt.show()
```

```
c:\users\admin\appdata\local\programs\python\python38\lib\site-packages\panda
s\plotting\_matplotlib\tools.py:331: MatplotlibDeprecationWarning:
The is_first_col function was deprecated in Matplotlib 3.4 and will be remove
d two minor releases later. Use ax.get_subplotspec().is_first_col() instead.
  if ax.is_first_col():
c:\users\admin\appdata\local\programs\python\python38\lib\site-packages\panda
s\plotting\_matplotlib\tools.py:331: MatplotlibDeprecationWarning:
The is_first_col function was deprecated in Matplotlib 3.4 and will be remove
d two minor releases later. Use ax.get_subplotspec().is_first_col() instead.
  if ax.is_first_col():
```



In [35]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, precision_s
core, recall_score, confusion_matrix, precision_recall_curve
from sklearn.preprocessing import RobustScaler
```

In [36]:
```
kp.head(1)
```

Out[36]:

	satisfaction	evaluation	projectCount	averageMonthlyHours	yearsAtCompany	workAccident
0	0.38	0.53	2	157	3	0

In [37]:
```
kp["Department"] = kp["Department"].astype('category').cat.codes
kp["salary"] = kp["salary"].astype('category').cat.codes
```

In [38]:
```
front = kp['turnover']
kp.drop(labels=['turnover'], axis=1,inplace = True)
kp.insert(0, 'turnover', front)

kp['int'] = 1
indep_var = ['satisfaction', 'evaluation', 'yearsAtCompany', 'int', 'turnover'
]
kp = kp[indep_var]
```

In [39]:
```
# Create train and test splits
target_name = 'turnover'
X = kp.drop('turnover', axis=1)

y=kp[target_name]
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.15, random
_state=123, stratify=y)

X_train.head()
```

Out[39]:

	satisfaction	evaluation	yearsAtCompany	int
9003	0.59	1.00	3	1
5697	0.81	0.98	2	1
10691	1.00	0.93	2	1
1884	0.87	0.91	5	1
13572	0.87	0.48	3	1

In [40]:
```
# Create base rate model
def base_rate_model(X) :
    y = np.zeros(X.shape[0])
    return y
```

In [41]:
```
#Check accuracy of base rate model
y_base_rate = base_rate_model(X_test)
from sklearn.metrics import accuracy_score
print ("Base rate accuracy is %2.2f" % accuracy_score(y_test, y_base_rate))
```

```
Base rate accuracy is 0.76
```

In [42]:
```
# Check accuracy of Logistic Model
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(penalty='l2', C=1)

model.fit(X_train, y_train)
print ("Logistic accuracy is %2.2f" % accuracy_score(y_test, model.predict(X_t
est)))
```

```
Logistic accuracy is 0.77
```

In [43]:
```
from sklearn import model_selection
```

In [44]:
```
kfold = model_selection.KFold(n_splits=10, random_state=7)
modelCV = LogisticRegression(class_weight = "balanced")
scoring = 'roc_auc'
results = model_selection.cross_val_score(modelCV, X_train, y_train, cv=kfold,
scoring=scoring)
print("AUC: %.3f (%.3f)" % (results.mean(), results.std()))
```

```
c:\users\admin\appdata\local\programs\python\python38\lib\site-packages\sklea
rn\model_selection\_split.py:293: FutureWarning: Setting a random_state has n
o effect since shuffle is False. This will raise an error in 0.24. You should
leave random_state to its default (None), or set shuffle=True.
  warnings.warn(

AUC: 0.793 (0.014)
```

In [45]:
```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import classification_report
```

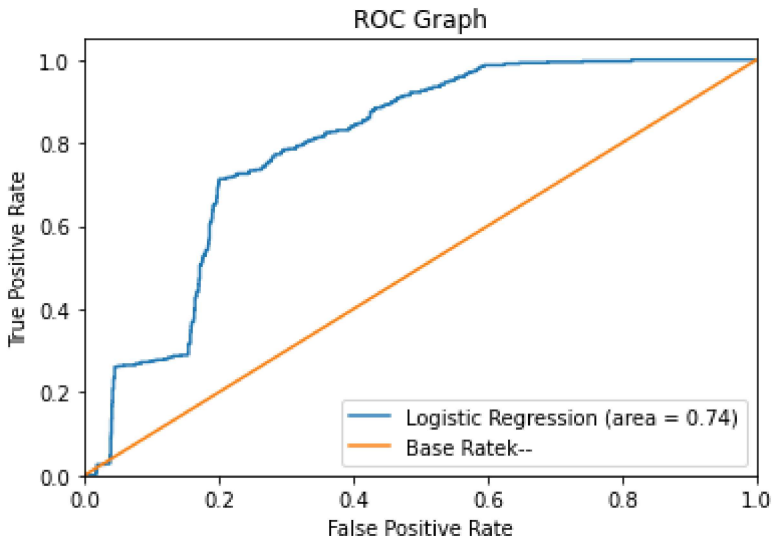
In [46]:
```
print ("---Base Model---")
base_roc_auc = roc_auc_score(y_test, base_rate_model(X_test))
print ("Base Rate AUC = %2.2f" % base_roc_auc)
print(classification_report(y_test, base_rate_model(X_test)))
```

```
---Base Model---
Base Rate AUC = 0.50
              precision    recall  f1-score   support

           0       0.76      1.00      0.86      1714
           1       0.00      0.00      0.00       536

    accuracy                           0.76      2250
   macro avg       0.38      0.50      0.43      2250
weighted avg       0.58      0.76      0.66      2250


c:\users\admin\appdata\local\programs\python\python38\lib\site-packages\sklea
rn\metrics\_classification.py:1221: UndefinedMetricWarning: Precision and F-s
core are ill-defined and being set to 0.0 in labels with no predicted sample
s. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [47]:
```
logis = LogisticRegression(class_weight = "balanced")
logis.fit(X_train, y_train)
print ("\n\n ---Logistic Model---")
logit_roc_auc = roc_auc_score(y_test, logis.predict(X_test))
print ("Logistic AUC = %2.2f" % logit_roc_auc)
print(classification_report(y_test, logis.predict(X_test)))
```

```
 ---Logistic Model---
Logistic AUC = 0.74
              precision    recall  f1-score   support

           0       0.90      0.76      0.82      1714
           1       0.48      0.73      0.58       536

    accuracy                           0.75      2250
   macro avg       0.69      0.74      0.70      2250
weighted avg       0.80      0.75      0.76      2250
```

In [48]:
```
from sklearn.metrics import roc_curve
fpr, tpr, thresholds = roc_curve(y_test, logis.predict_proba(X_test)[:,1])
plt.figure()

# Plot Logistic Regression ROC
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)

# Plot Base Rate ROC
plt.plot([0,1], [0,1],label='Base Rate' 'k--')

plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Graph')
plt.legend(loc="lower right")
plt.show()
```



K-Means Clustering of Employee Turnover

In [49]:
```
from sklearn.cluster import KMeans

# Graph and create 3 clusters of Employee Turnover
kmeans = KMeans(n_clusters=3,random_state=2)
kmeans.fit(kp[kp.turnover==1][["satisfaction","evaluation"]])

kmeans_colors = ['red' if c == 0 else 'blue' if c == 2 else 'green' for c in k
means.labels_]

fig = plt.figure(figsize=(10, 6))
plt.scatter(x="satisfaction",y="evaluation", data=kp[kp.turnover==1],
            alpha=0.25,color = kmeans_colors)
plt.xlabel("Satisfaction")
plt.ylabel("Evaluation")
plt.scatter(x=kmeans.cluster_centers_[:,0],y=kmeans.cluster_centers_[:,1],colo
r="black",marker="X",s=100)
plt.title("Clusters of Employee Turnover")
plt.show()
```

Conclusion:-

Cluster 1 (Blue): Hard-working and Sad Employees.

Cluster 2 (Red): Bad and Sad Employee.

Cluster 3 (Green): Hard-working and Happy Employee

Confusion matrix

In [50]:
```
confusion_matrix(y_test, model.predict(X_test))
```

Out[50]:
```
array([[1581,  133],
       [ 390,  146]], dtype=int64)
```

Classification Error:

In [51]:
```
print(1 - accuracy_score(y_test,model.predict(X_test)))
```

```
0.23244444444444445
```

Sensitivity/True Positive Rate/Recall Score/F1-Score/Precision

In [52]:
```
print(recall_score(y_test, model.predict(X_test)))
```

```
0.27238805970149255
```

In [53]:
```
vin=model.predict(X_test)
```

In [54]:
```
print(recall_score(y_test, vin,average=None))
```

```
[0.92240373 0.27238806]
```

In [55]:
```
print(precision_score(y_test, vin,average=None))
```

```
[0.8021309  0.52329749]
```

In [56]:
```
from sklearn.metrics import  f1_score
print(f1_score(y_test, vin,average=None))
```

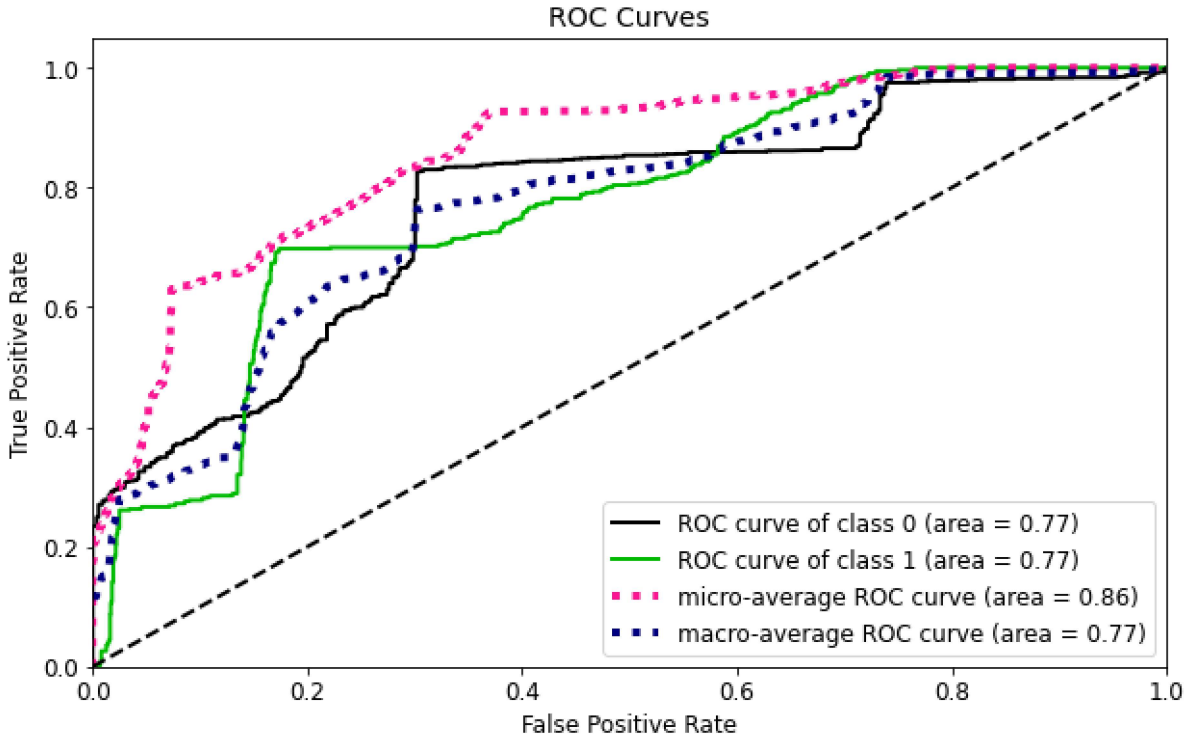
```
[0.85807327 0.35828221]
```

In [57]:
```
pip install scikit-plot
```

```
Requirement already satisfied: scikit-plot in c:\users\admin\appdata\local\pr
ograms\python\python38\lib\site-packages (0.3.7)
Requirement already satisfied: matplotlib>=1.4.0 in c:\users\admin\appdata\lo
cal\programs\python\python38\lib\site-packages (from scikit-plot) (3.4.2)
Requirement already satisfied: joblib>=0.10 in c:\users\admin\appdata\local\p
rograms\python\python38\lib\site-packages (from scikit-plot) (0.14.1)
Requirement already satisfied: scikit-learn>=0.18 in c:\users\admin\appdata\l
ocal\programs\python\python38\lib\site-packages (from scikit-plot) (0.23.2)
Requirement already satisfied: scipy>=0.9 in c:\users\admin\appdata\local\pro
grams\python\python38\lib\site-packages (from scikit-plot) (1.4.1)
Requirement already satisfied: numpy>=1.16 in c:\users\admin\appdata\local\pr
ograms\python\python38\lib\site-packages (from matplotlib>=1.4.0->scikit-plo
t) (1.18.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\admin\appdata\lo
cal\programs\python\python38\lib\site-packages (from matplotlib>=1.4.0->sciki
t-plot) (1.1.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\admin\appdata\local
\programs\python\python38\lib\site-packages (from matplotlib>=1.4.0->scikit-p
lot) (7.2.0)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\admin\appdata\loc
al\programs\python\python38\lib\site-packages (from matplotlib>=1.4.0->scikit
-plot) (2.4.6)
Requirement already satisfied: cycler>=0.10 in c:\users\admin\appdata\local\p
rograms\python\python38\lib\site-packages (from matplotlib>=1.4.0->scikit-plo
t) (0.10.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\admin\appdata
\local\programs\python\python38\lib\site-packages (from matplotlib>=1.4.0->sc
ikit-plot) (2.8.1)
Requirement already satisfied: six in c:\users\admin\appdata\local\programs\p
ython\python38\lib\site-packages (from cycler>=0.10->matplotlib>=1.4.0->sciki
t-plot) (1.14.0)
Requirement already satisfied: setuptools in c:\users\admin\appdata\local\pro
grams\python\python38\lib\site-packages (from kiwisolver>=1.0.1->matplotlib>=
1.4.0->scikit-plot) (41.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\admin\appdata
\local\programs\python\python38\lib\site-packages (from scikit-learn>=0.18->s
cikit-plot) (2.1.0)
Note: you may need to restart the kernel to use updated packages.

WARNING: Ignoring invalid distribution -atplotlib (c:\users\admin\appdata\loc
al\programs\python\python38\lib\site-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\admin\appdata\loc
al\programs\python\python38\lib\site-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\admin\appdata\loc
al\programs\python\python38\lib\site-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\admin\appdata\loc
al\programs\python\python38\lib\site-packages)
WARNING: Ignoring invalid distribution -atplotlib (c:\users\admin\appdata\loc
al\programs\python\python38\lib\site-packages)
```

In [60]:
```
import scikitplot as skplt
```

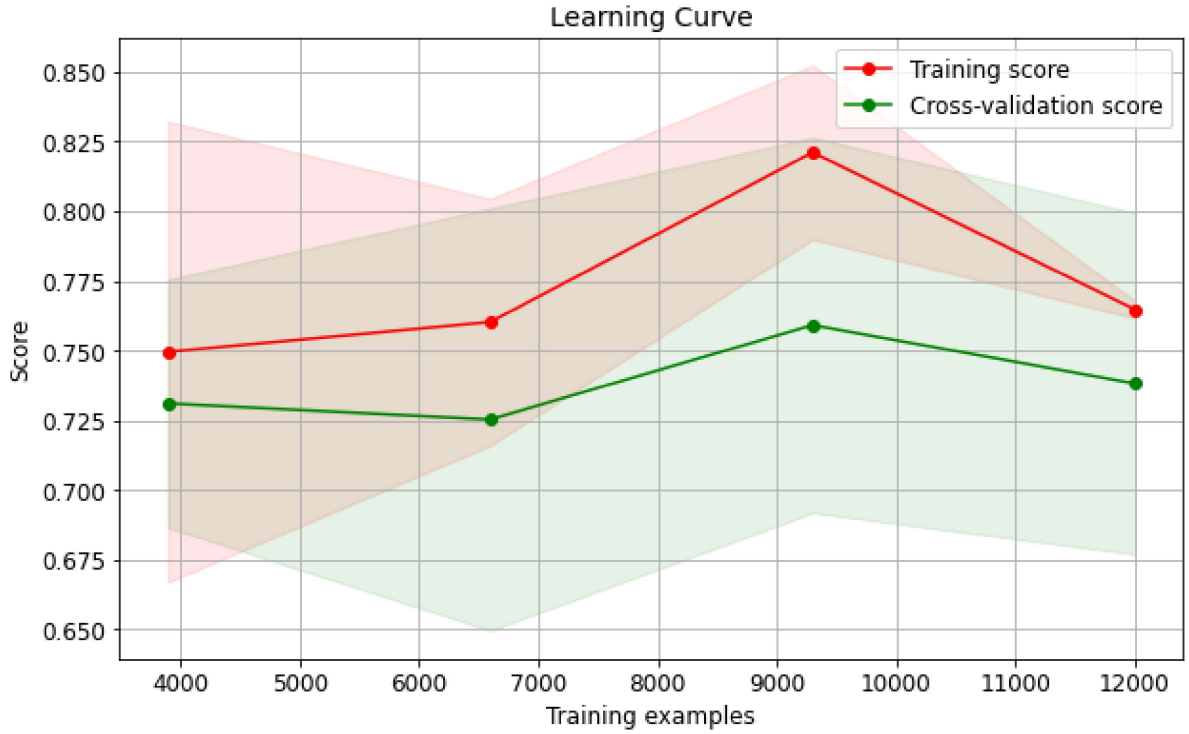
In [61]:
```
y_probas = model.predict_proba(X_test)
skplt.metrics.plot_roc(y_test,y_probas,figsize=(10,6),title_fontsize=14,text_f
ontsize=12)
plt.show()
```



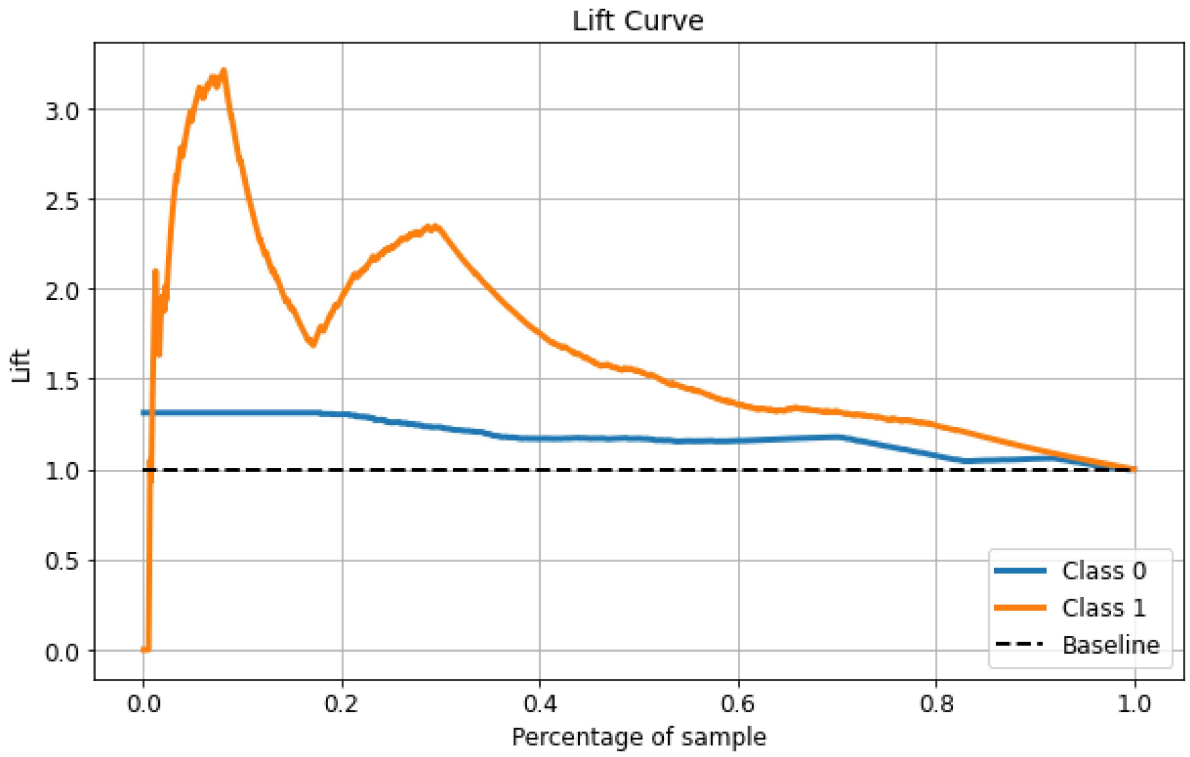
In [62]:
```
skplt.estimators.plot_learning_curve(model, X,y,figsize=(10,6),title_fontsize=
14,text_fontsize=12)
plt.show()
```

```
c:\users\admin\appdata\local\programs\python\python38\lib\site-packages\sklea
rn\model_selection\_validation.py:548: FitFailedWarning: Estimator fit faile
d. The score on this train-test partition for these parameters will be set to
nan. Details:
Traceback (most recent call last):
  File "c:\users\admin\appdata\local\programs\python\python38\lib\site-packag
es\sklearn\model_selection\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "c:\users\admin\appdata\local\programs\python\python38\lib\site-packag
es\sklearn\linear_model\_logistic.py", line 1372, in fit
    raise ValueError("This solver needs samples of at least 2 classes"
ValueError: This solver needs samples of at least 2 classes in the data, but
the data contains only one class: 1

  warnings.warn("Estimator fit failed. The score on this train-test"
c:\users\admin\appdata\local\programs\python\python38\lib\site-packages\sklea
rn\model_selection\_validation.py:548: FitFailedWarning: Estimator fit faile
d. The score on this train-test partition for these parameters will be set to
nan. Details:
Traceback (most recent call last):
  File "c:\users\admin\appdata\local\programs\python\python38\lib\site-packag
es\sklearn\model_selection\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "c:\users\admin\appdata\local\programs\python\python38\lib\site-packag
es\sklearn\linear_model\_logistic.py", line 1372, in fit
    raise ValueError("This solver needs samples of at least 2 classes"
ValueError: This solver needs samples of at least 2 classes in the data, but
the data contains only one class: 1

  warnings.warn("Estimator fit failed. The score on this train-test"
c:\users\admin\appdata\local\programs\python\python38\lib\site-packages\sklea
rn\model_selection\_validation.py:548: FitFailedWarning: Estimator fit faile
d. The score on this train-test partition for these parameters will be set to
nan. Details:
Traceback (most recent call last):
  File "c:\users\admin\appdata\local\programs\python\python38\lib\site-packag
es\sklearn\model_selection\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "c:\users\admin\appdata\local\programs\python\python38\lib\site-packag
es\sklearn\linear_model\_logistic.py", line 1372, in fit
    raise ValueError("This solver needs samples of at least 2 classes"
ValueError: This solver needs samples of at least 2 classes in the data, but
the data contains only one class: 1

  warnings.warn("Estimator fit failed. The score on this train-test"
c:\users\admin\appdata\local\programs\python\python38\lib\site-packages\sklea
rn\model_selection\_validation.py:548: FitFailedWarning: Estimator fit faile
d. The score on this train-test partition for these parameters will be set to
nan. Details:
Traceback (most recent call last):
  File "c:\users\admin\appdata\local\programs\python\python38\lib\site-packag
es\sklearn\model_selection\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "c:\users\admin\appdata\local\programs\python\python38\lib\site-packag
es\sklearn\linear_model\_logistic.py", line 1372, in fit
    raise ValueError("This solver needs samples of at least 2 classes"
ValueError: This solver needs samples of at least 2 classes in the data, but
```

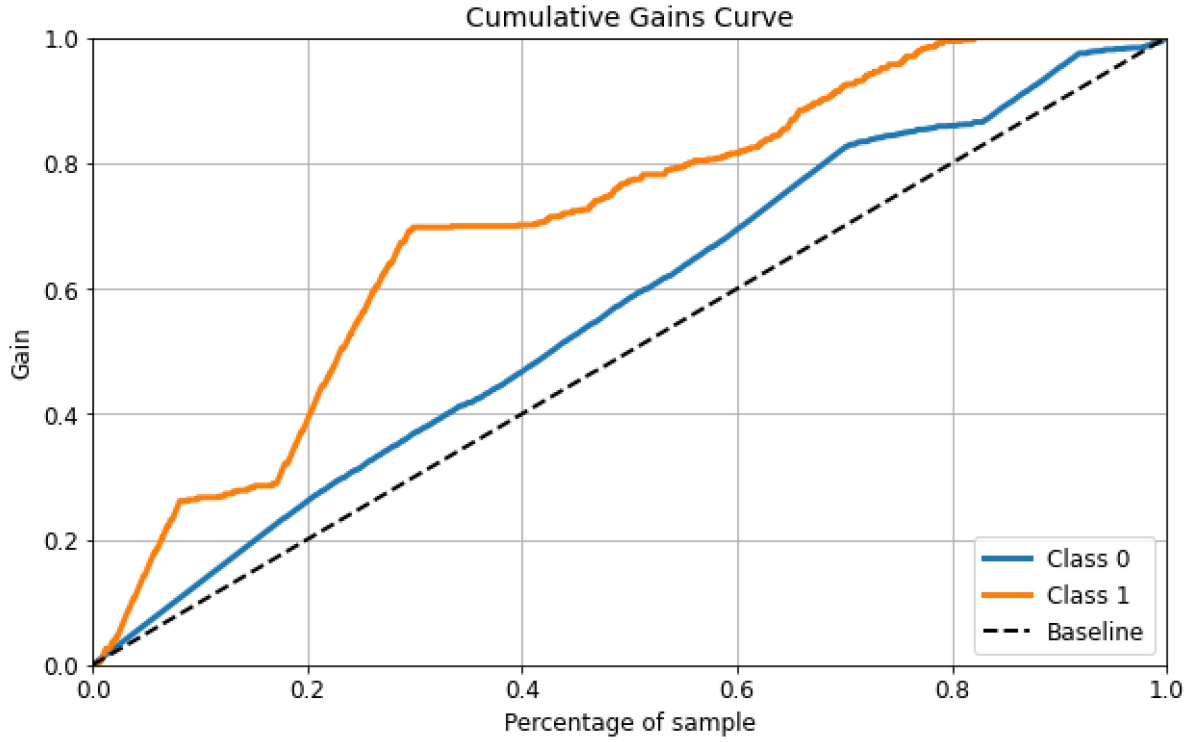
```
the data contains only one class: 1

  warnings.warn("Estimator fit failed. The score on this train-test"
c:\users\admin\appdata\local\programs\python\python38\lib\site-packages\sklea
rn\model_selection\_validation.py:548: FitFailedWarning: Estimator fit faile
d. The score on this train-test partition for these parameters will be set to
nan. Details:
Traceback (most recent call last):
  File "c:\users\admin\appdata\local\programs\python\python38\lib\site-packag
es\sklearn\model_selection\_validation.py", line 531, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "c:\users\admin\appdata\local\programs\python\python38\lib\site-packag
es\sklearn\linear_model\_logistic.py", line 1372, in fit
    raise ValueError("This solver needs samples of at least 2 classes"
ValueError: This solver needs samples of at least 2 classes in the data, but
the data contains only one class: 1

  warnings.warn("Estimator fit failed. The score on this train-test"
```



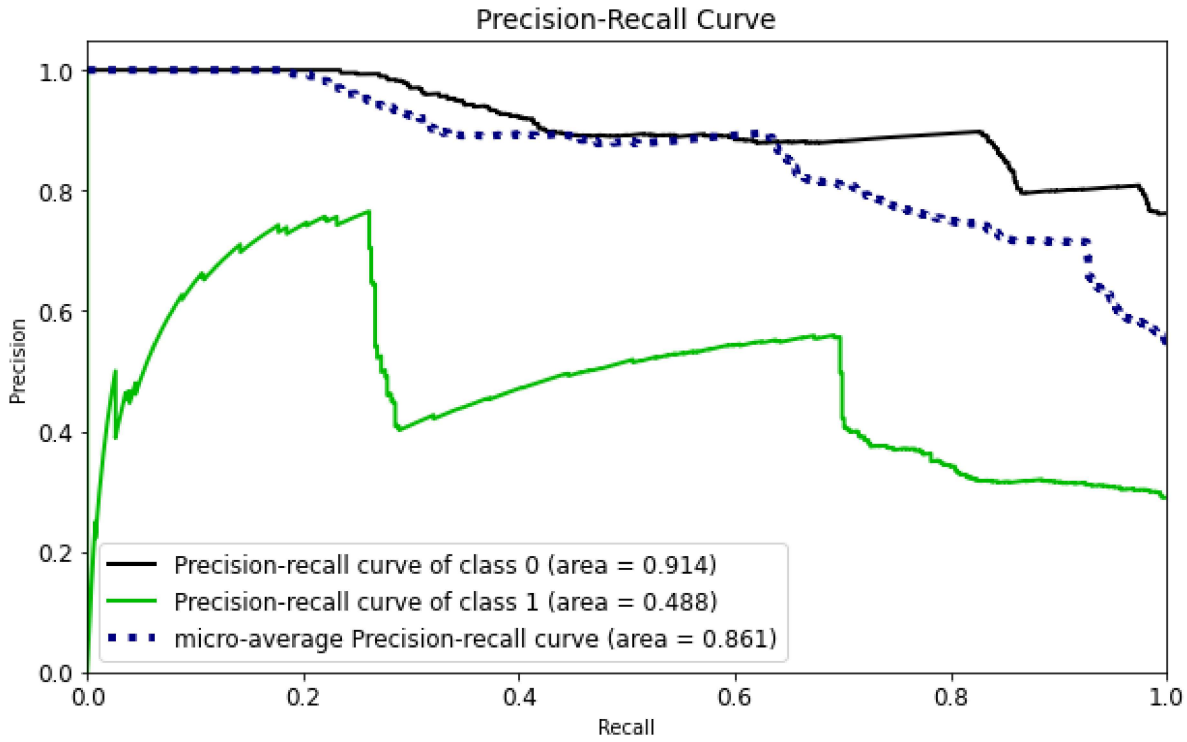
In [63]:
```
skplt.metrics.plot_lift_curve(y_test,y_probas,figsize=(10,6),title_fontsize=14
,text_fontsize=12)
plt.show()
```



In [64]:
```
skplt.metrics.plot_cumulative_gain(y_test,y_probas,figsize=(10,6),title_fontsi
ze=14,text_fontsize=12)
plt.show()
```



In [65]:
```
skplt.metrics.plot_precision_recall(y_test,y_probas,figsize=(10,6),title_fonts
ize=14,text_fontsize=12)
plt.show()
```



Confusion matrix

In [66]:
```
confusion_matrix(y_test, vin)
```

Out[66]:
```
array([[1581,  133],
       [ 390,  146]], dtype=int64)
```

THANK YOU