Employee Churn Prediction

Project By : PRASAD JADHAV

In [41]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import OrdinalEncoder


from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier

import warnings

import pickle
```

In [2]:
```
warnings.filterwarnings('ignore')
```

In [3]:
```
dataset = pd.read_csv('HR_Dataset.csv')
```

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

Out[4]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company
0	0.38	0.53	2	157	3
1	0.80	0.86	5	262	6
2	0.11	0.88	7	272	4
3	0.72	0.87	5	223	5
4	0.37	0.52	2	159	3

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

Out[5]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_comp
14994	0.40	0.57	2	151	
14995	0.37	0.48	2	160	
14996	0.37	0.53	2	143	
14997	0.11	0.96	6	280	
14998	0.37	0.52	2	158	

In [6]:
```
dataset.shape
```

Out[6]:
```
(14999, 10)
```

In [7]:
```
print('Number of Rows:', dataset.shape[0])
print('Number of Columns:', dataset.shape[1])
```

```
Number of Rows: 14999
Number of Columns: 10
```

In [8]:
```
dataset.rename(columns={'Departments ':'departments'}, inplace=True)
```

In [9]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   satisfaction_level     14999 non-null  float64
 1   last_evaluation        14999 non-null  float64
 2   number_project         14999 non-null  int64  
 3   average_montly_hours   14999 non-null  int64  
 4   time_spend_company     14999 non-null  int64  
 5   Work_accident          14999 non-null  int64  
 6   left                   14999 non-null  int64  
 7   promotion_last_5years  14999 non-null  int64  
 8   departments            14999 non-null  object 
 9   salary                 14999 non-null  object 
dtypes: float64(2), int64(6), object(2)
memory usage: 1.1+ MB
```

In [10]:
```
dataset.isnull().sum()
```

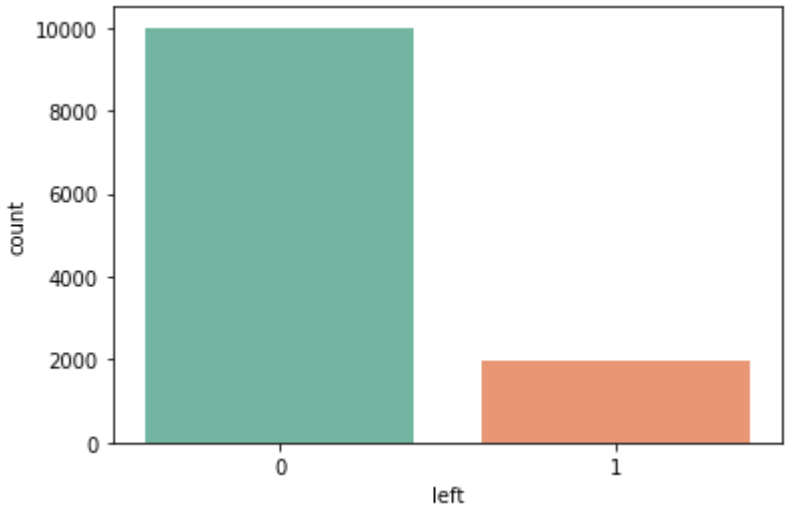
Out[10]:
```
satisfaction_level       0
last_evaluation          0
number_project           0
average_montly_hours     0
time_spend_company       0
Work_accident            0
left                     0
promotion_last_5years    0
departments              0
salary                   0
dtype: int64
```

In [11]:
```
dataset.duplicated().sum()
```

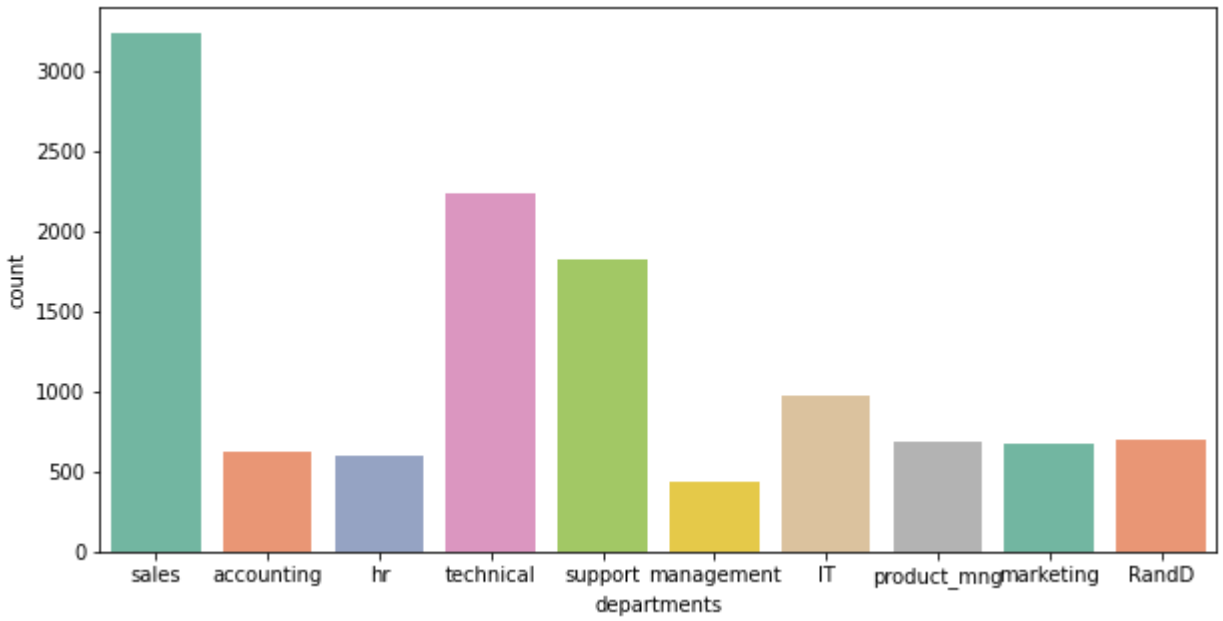
Out[11]:
```
3008
```

In [12]:
```
dataset = dataset.drop_duplicates()
```

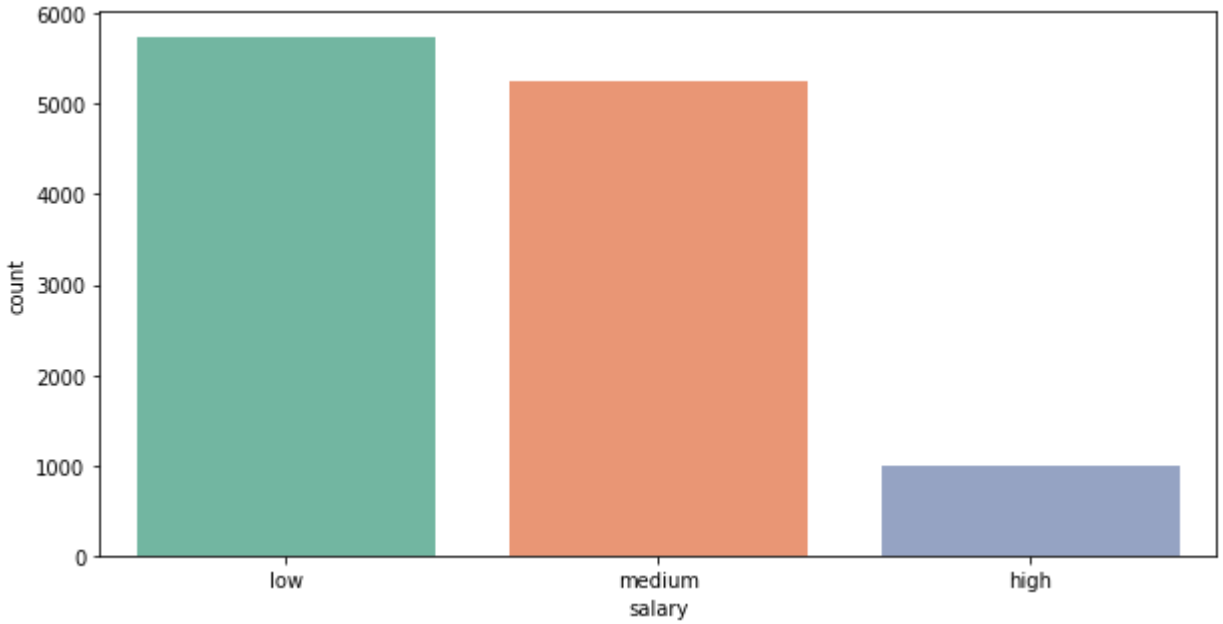
In [13]:
```
sns.countplot(dataset['left'], palette='Set2')
plt.show()
```



In [14]:
```
plt.figure(figsize=(10,5))
sns.countplot(dataset['departments'], palette='Set2')
plt.show()
```



In [15]:
```
plt.figure(figsize=(10,5))
sns.countplot(dataset['salary'], palette='Set2')
plt.show()
```

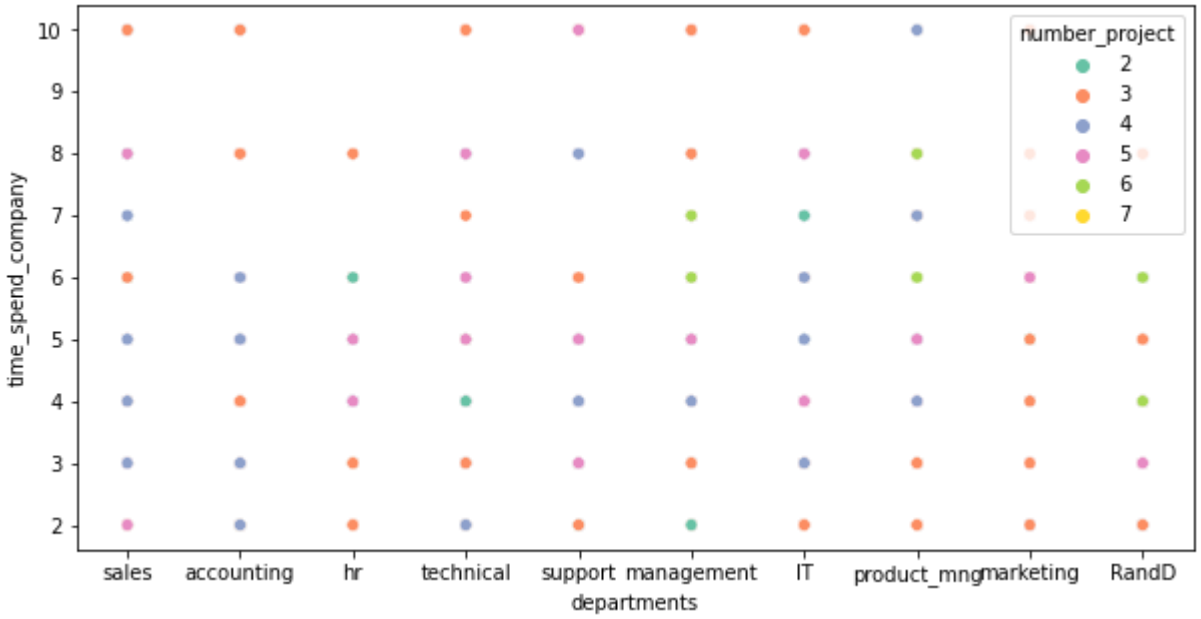


In [16]:
```
dataset.head()
```

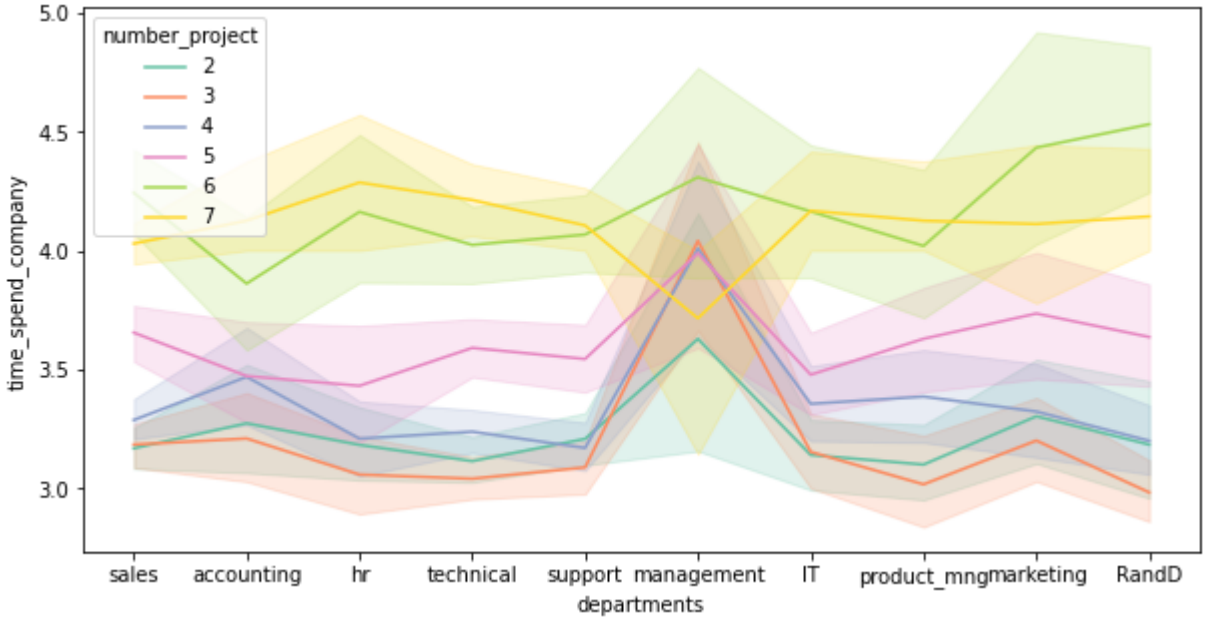
Out[16]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company
0	0.38	0.53	2	157	3
1	0.80	0.86	5	262	6
2	0.11	0.88	7	272	4
3	0.72	0.87	5	223	5
4	0.37	0.52	2	159	3

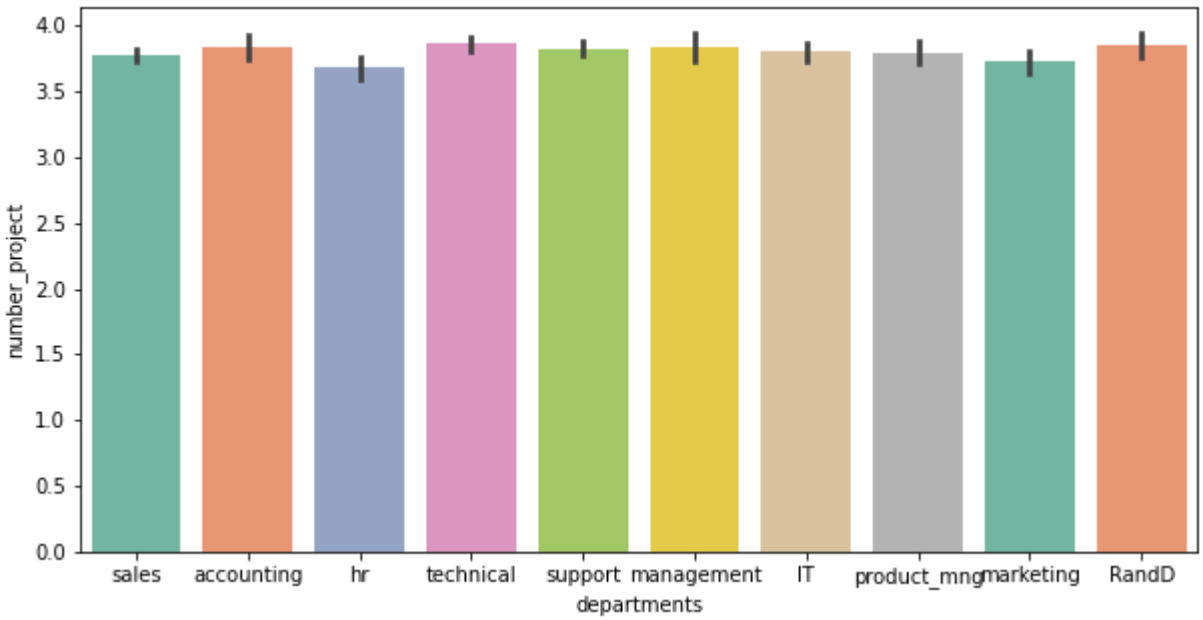
In [17]:
```
plt.figure(figsize=(10,5))
sns.scatterplot(dataset['departments'], dataset['time_spend_company'], hue=da
plt.show()
```



In [18]:
```
plt.figure(figsize=(10,5))
sns.lineplot(dataset['departments'], dataset['time_spend_company'], hue=datas
plt.show()
```



In [22]:
```
plt.figure(figsize=(10,5))
sns.barplot(dataset['departments'], dataset['number_project'], palette='Set2
plt.show()
```



In [23]:
```
X = dataset.drop(columns=['left'])
y = dataset['left']
```

In [25]:
```
preprocessor = ColumnTransformer(transformers=[
    ('num', StandardScaler(),['satisfaction_level',
                              'last_evaluation',
                              'number_project',
                              'average_montly_hours',
                              'time_spend_company',
                              'Work_accident', 'promotion_last_5years']),
    ('nominal', OneHotEncoder(),['departments']),
    ('ordinal', OrdinalEncoder(),['salary'])


], remainder='passthrough')
```

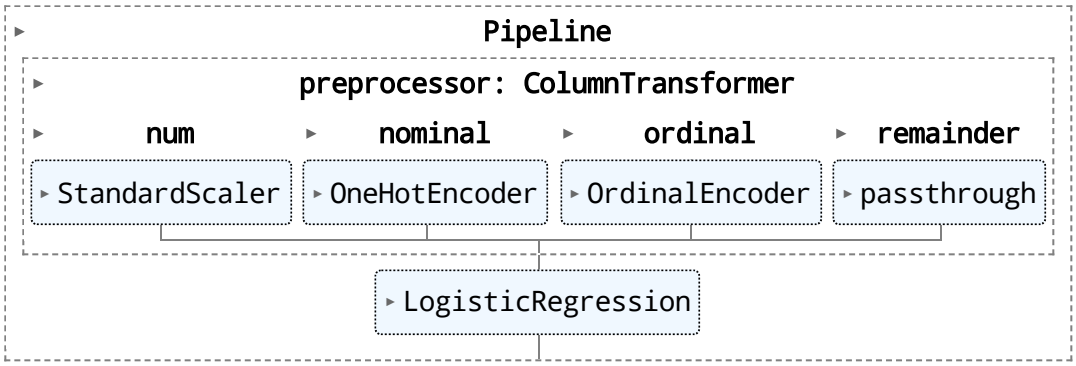
In [26]:
```
pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('model', LogisticRegression())

])
```

In [27]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, ran
```

In [28]:
```
pipeline.fit(X_train, y_train)
```

Out[28]:



In [29]:
```
y_pred = pipeline.predict(X_test)
```

In [30]:
```
accuracy_score(y_test, y_pred)
```

Out[30]:
```
0.8370154230929554
```

In [31]:
```
precision_score(y_test, y_pred)
```

Out[31]:
```
0.5209580838323353
```

In [32]:
```
recall_score(y_test, y_pred)
```

Out[32]:
```
0.2185929648241206
```

In [33]:
```
def model_scorer(model_name,model):

    output=[]

    output.append(model_name)

    pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('model', model)])

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,

    pipeline.fit(X_train, y_train)

    y_pred = pipeline.predict(X_test)

    output.append(accuracy_score(y_test, y_pred))

    output.append(precision_score(y_test, y_pred))

    output.append(recall_score(y_test, y_pred))

    return output
```

In [35]:
```
model_dict={

    'log': LogisticRegression(),
    'decision_tree': DecisionTreeClassifier(),
    'random_forest': RandomForestClassifier(),
    'xgb': XGBClassifier(),
    'ad_bost': AdaBoostClassifier(),
    'gradi_bost': GradientBoostingClassifier()

}
```

In [36]:
```
model_output=[]
for model_name,model in model_dict.items():
    model_output.append(model_scorer(model_name, model))
```

In [37]:
```
model_output
```

Out[37]:
```
[['log', 0.8370154230929554, 0.5209580838323353, 0.2185929648241206],
 ['decision_tree', 0.9699874947894956, 0.8975609756097561, 0.924623115577889
5],
 ['random_forest', 0.9862442684451855, 0.9892761394101877, 0.927135678391959
8],
 ['xgb', 0.9824927052938724, 0.9635416666666666, 0.9296482412060302],
 ['ad_bost', 0.9662359316381826, 0.8992443324937027, 0.8969849246231156],
 ['gradi_bost', 0.9795748228428511, 0.9462915601023018, 0.9296482412060302]]
```

In [38]:
```
sample = pd.DataFrame({

   'satisfaction_level':0.38,
   'last_evaluation':0.53,
    'number_project':2,
    'average_montly_hours':157,
    'time_spend_company':3,
    'Work_accident':0,
    'promotion_last_5years':0,
    'departments':'sales',
    'salary':'low'


},index=[0])
```

In [40]:
```
result = pipeline.predict(sample)

if result == 1:
    print('Employee May Leave the Organization')
else:
    print('Employee May Stay with the Organization')
```

```
Employee May Stay with the Organization
```

In [42]:
```
pickle.dump(pipeline,open('employee_churn_predictor.pkl','wb'))
```

In [43]:
```
model = pickle.load(open('employee_churn_predictor.pkl','rb'))
```

In [44]:
```
result = model.predict(sample)

if result == 1:
    print('Employee May Leave the Organization')
else:
    print('Employee May Stay with the Organization')
```

```
Employee May Stay with the Organization
```

In [ ]:
```
from tkinter import *

def show_entry():

    model = pickle.load(open('employee_churn_predictor.pkl','rb'))

    p1 = float(e1.get())
    p2 = float(e2.get())
    p3 = float(e3.get())
    p4 = float(e4.get())
    p5 = float(e5.get())
    p6 = float(e6.get())
    p7 = float(e7.get())
```

```
    p8 = str(clicked.get())
    p9 = str(clicked1.get())

    sample = pd.DataFrame({

        'satisfaction_level': [p1],
        'last_evaluation': [p2],
        'number_project': [p3],
        'average_montly_hours': [p4],
        'time_spend_company': [p5],
        'Work_accident': [p6],
        'promotion_last_5years': [p7],
        'departments': [p8],
        'salary': [p9]
    })

    result = model.predict(sample)
    print(result)

    if result == 1:
        Label(master, text='Employee May Leave the Organization').grid(row=31
    else:
        Label(master, text='Employee May Stay with the Organization').grid(ro



master =Tk()
master.title('Employee Churn Prediction Using Machine Learning')
label = Label(master, text='Status of Employee Churn', bg='black',
              fg='white').grid(row=0, columnspan=2)

Label(master, text='Employee Satisfaction Level').grid(row=1)
Label(master, text='Last Evaluation Score').grid(row=2)
Label(master, text='Number of Projects Assigned to').grid(row=3)
Label(master, text='Average Monthly Kours Worked').grid(row=4)
Label(master, text='Time Spent at the Company').grid(row=5)
Label(master, text='Whether they have had a Work Accident (1/Yes, 0/No)').gri
Label(master, text='Whether they have had a Promotion in the Last 5 Years (1/
Label(master, text='Department Name').grid(row=8)
Label(master, text='Salary Category').grid(row=9)


clicked = StringVar()
options = ['sales', 'technical', 'support', 'IT', 'product_mng', 'marketing',
       'RandD', 'accounting', 'hr', 'management']

clicked1 = StringVar()
options1 = ['low', 'medium', 'high']

e1 = Entry(master)
e2 = Entry(master)
e3 = Entry(master)
e4 = Entry(master)
e5 = Entry(master)
e6 = Entry(master)
e7 = Entry(master)
# e8 = Entry(master)
e8 = OptionMenu(master, clicked, *options )
e8.configure(width=15)
```

```
# e9 = Entry(master)
e9 = OptionMenu(master, clicked1, *options1 )
e9.configure(width=15)


e1.grid(row=1, column=1)
e2.grid(row=2, column=1)
e3.grid(row=3, column=1)
e4.grid(row=4, column=1)
e5.grid(row=5, column=1)
e6.grid(row=6, column=1)
e7.grid(row=7, column=1)
e8.grid(row=8, column=1)
e9.grid(row=9, column=1)


Button(master, text='Predict', command=show_entry).grid()

mainloop()
```