Abstract:

The aim of this study is to predict the rate of employees' early turn-over in terms of employee stress, role clarity, employee satisfaction, turn-over, and interaction effect between the variables. Data was collected from 5074 employees within the first six months they start to work at the company. This study applies linear and non-linear models to find out which model predicts the dependent variable.

The objectives of this study are two-folds. First, this study applies linear logistic regression to classify rate of employees' turn-over into a binary variable: employees who stay within the six months, and employees who leave the company. Second, non-linear model- neural network is applied to predict the rate of employees' early turn-over. The results indicate that neural network is the best model to predict the employees' early turn-over compared to the linear logistic regression.

Appendix I

Python code for logistic regression

```
import warnings
from sklearn.metrics import roc_auc_score
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score
from sklearn import model_selection
from sklearn.model_selection import train_test_split, cross_val_score,check_cv, permutation_test_score
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
warnings.filterwarnings('ignore')
file_missing=pd.read_csv('c://users/atieh/Desktop/Newhired_Python.csv')
file=file_missing.dropna()
filex=file.iloc[:,0:20]
filey=file.iloc[:,[20]]
trainx=filex.iloc[500:,:]
scaler=MinMaxScaler()
scaler.fit(trainx)
trainx=scaler.transform(trainx)
trainy=filey.iloc[500:,:]
seed=1234
trainx, testx, trainy, testy= model_selection. train_test_split(filex,filey,test_size=.3, random_state=seed)
model=LogisticRegression()
```

```
model.fit(trainx,trainy)
print("model train", model)
testx=filex.iloc[:500,:]
scaler=MinMaxScaler()
scaler.fit(testx)
testx=scaler.transform(testx)
testy=filey.iloc[:500,:]
prediction=model.predict(testx)
print ("prediction", prediction)
print ("score", confusion_matrix (testy, prediction))
print("accuracy", model_selection.cross_val_score(model,trainx,trainy,scoring='accuracy'))
print("precision", model_selection.cross_val_score(model,testx,testy,scoring='precision'))
print("class matrix", classification_report (testy,prediction))
results=model_selection.cross_val_score(model,testx,testy)
print("proba",model.predict_proba(testx)[:,0])
print ("results for test set ",results)
auc= roc_auc_score(testy, prediction)
print("auc",auc)
```

Output

```
model train LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
        intercept_scaling=1, max_iter=100, multi_class='warn',
        n_jobs=None, penalty='l2', random_state=None, solver='warn',
        tol=0.0001, verbose=0, warm_start=False)
score [[459   0]  [ 41   0]]
accuracy [0.88438819 0.88334742 0.88334742]
precision [0. 0. 0.]
```

class matrix	precision	recall	f1-score	support
0	0.92	1.00	0.96	459
1	0.00	0.00	0.00	41
micro avg	0.92	0.92	0.92	500
macro avg	0.46	0.50	0.48	500
weighted avg	0.84	0.92	0.88	500

auc 0.5

Process finished with exit code 0

Python code using neural network

```
import warnings
from sklearn.metrics import roc_auc_score
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import MinMaxScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
from sklearn import model_selection
from sklearn.model_selection import train_test_split, cross_val_score,check_cv,
permutation_test_score
from sklearn.metrics import classification_report, confusion_matrix
warnings.filterwarnings('ignore')
file_missing=pd.read_csv('c://users/atieh/Desktop/Newhired_Python.csv')
file=file_missing.dropna()
filex=file.iloc[:,0:20]
filey=file.iloc[:,[20]]
trainx=filex.iloc[500:,:]
scaler=MinMaxScaler()
scaler.fit(trainx)
trainx=scaler.transform(trainx)
trainy=filey.iloc[500:,:]
seed=10
kfold=model_selection.KFold(n_splits=10,random_state=seed)
mlp=MLPClassifier (hidden_layer_sizes = (10,10), activation='tanh', max_iter=5000,
alpha=0.0001, early_stopping=False, solver='lbfgs', shuffle= True, learning_rate= 'adaptive',
random_state=1)
mlp.fit(trainx,trainy)
testx=filex.iloc[:500,:]
```

```
scaler=MinMaxScaler()
scaler.fit(testx)
testx=scaler.transform(testx)
testy=filey.iloc[:500,:]
prediction=mlp.predict(testx)
print("val scores", mlp.loss)
print("score",confusion_matrix(testy, prediction))
print ("class matrix",classification_report(testy, prediction))
scoring='accuracy'
results=model_selection.cross_val_score( mlp, testx, testy, cv= kfold)
print ("results for kfold testx and test y",results)
```

Output

```
score [[447  12]  [ 40   1]]
class matrix              precision    recall   f1-score   support
             0                0.92        0.97      0.95       459
             1                0.08        0.02      0.04        41

   micro avg        0.90      0.90        0.90       500
   macro avg        0.50      0.50        0.49       500
weighted avg        0.85      0.90        0.87       500

results for kfold testx and test y [0.92 0.92 0.84 0.84 0.86 0.86 0.9  0.86 0.88 0.8 ]

Process finished with exit code 0
```