

Employee Attrition Predict

Sinir Ağları ve Makine Öğrenmesi

Burak Yavaş-190701111 | Ahmet Emin Yalçınkaya-190701107

Şirketler genellikle her yıl yeni çalışanları eğitmek için büyük miktarda para harcarlar. Bir ciro olduğunda, işe alma ve eğitim sürecinden geçmek maliyetlidir. Bu nedenle çalışan bağlılığı, yönetim ekibinin dikkat etmesi gereken önemli konulardan biridir. Çalışanların yıpranmasına neden olan faktörleri ortaya çıkarmalı ve bu faktörleri ortadan kaldırarak şirket için gerekli olan verimliliği ve sürekliliği artırmak oldukça önemlidir.

Yıpranmaya neden olan faktörleri daha iyi keşfetmek ve karşılaştırmak için kullanıcıların farklı değişkenleri filtrelenmesini ve işlenmesini sağlayacağız.

▼ Süreçler

Verilere genel bakış

Kod yapısına genel bakış

Tablolara genel bakış

Sonuç

Verilere Genel Bakış

```
def dataFile():  
    df = pd.read_csv("../BigProject/Data/dataset.csv")  
    return df  
data = dataFile()  
data.columns
```

Ön Hazırlık

dataFile fonksiyonu içerisinde data okundu ve features(columns) bastırıldı.

```
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
      'DistanceFromHome', 'Education', 'EducationField',
      'EnvironmentSatisfaction', 'Gender', 'HourlyRate', 'JobInvolvement',
      'JobLevel', 'JobRole', 'JobSatisfaction', 'MaritalStatus',
      'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked', 'OverTime',
      'PercentSalaryHike', 'PerformanceRating', 'RelationshipSatisfaction',
      'StockOptionLevel', 'TotalWorkingYears', 'TrainingTimesLastYear',
      'WorkLifeBalance', 'YearsAtCompany', 'YearsInCurrentRole',
      'YearsSinceLastPromotion', 'YearsWithCurrManager'],
      dtype='object')
```

```
data.info()
```

```
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

```
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   Age                    1470 non-null   int64
1   Attrition              1470 non-null   object
2   BusinessTravel         1470 non-null   object
3   DailyRate              1470 non-null   int64
```

```
data = data.drop(["EmployeeCount", "EmployeeNumber", "Over18", "StandardHours", "Attrition_numerical"], axis=1)
```

- data Tipleri , sayıları hakkında veriler

- Bazı özellikler işlenebilir bir değere sahip değildir bu sebeple verilerin kaldırılması gerekmektedir.

Tablolara Ve Koda Bakış

```
def pearsonCorr(attrition):

    target_map = {'Yes': 1, 'No': 0}

    attrition["Attrition_numerical"] = attrition["Attrition"].apply(
        lambda x: target_map[x])

    numerical = [u'Age', u'DailyRate', u'DistanceFromHome',
                 u'Education', u'EmployeeNumber', u'EnvironmentSati
sfaction',
                 u'HourlyRate', u'JobInvolvement', u'JobLevel', u'J
```

Pearson Korelasyon Katsayısı

Pearson korelasyon katsayısı (r), doğrusal bir korelasyonu ölçmenin en

```

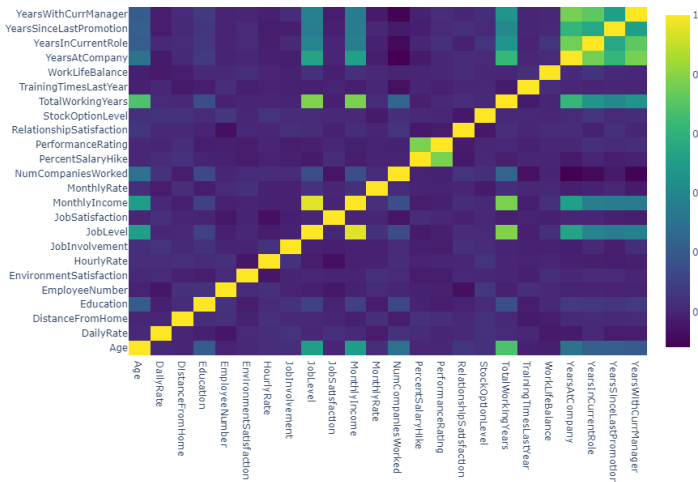
obSatisfaction',
    u'MonthlyIncome', u'MonthlyRate', u'NumCompaniesWo
rked',
    u'PercentSalaryHike', u'PerformanceRating', u'Rela
tionshipSatisfaction',
    u'StockOptionLevel', u'TotalWorkingYears',
    u'TrainingTimesLastYear', u'WorkLifeBalance', u'Ye
arsAtCompany',
    u'YearsInCurrentRole', u'YearsSinceLastPromotion',
u'YearsWithCurrManager']
data = [
    go.Heatmap(
        z = attrition[numerical].astype(float).corr().values,
        x = attrition[numerical].columns.values, y = attrition[numerical].
columns.values,
        colorscale = 'Viridis',
        reversescale = False,
        opacity=1.0
    )
]

layout = go.Layout(
    title='Pearson Korelasyonu',
    xaxis=dict(ticks='', nticks=36),
    yaxis=dict(ticks=''),
    width=900, height=700,
)

fig = go.Figure(data=data, layout=layout)
py.plot(fig, filename='labelled-map')

```

Pearson Korelasyonu



```

def analyseData(attrition):

    f, axes = plt.subplots(3, 3, figsize=(10, 8), sharex=False, sha
rey=False)

    s = np.linspace(0, 3, 10)

    cmap = sns.cubehelix_palette(start=0.0, light=1, as_cmap=True)
    x = attrition['Age'].values
    y = attrition['TotalWorkingYears'].values
    sns.kdeplot(x, y, cmap=cmap, shade=True, cut=5, ax=axes[0, 0])
    axes[0, 0].set(title='Age against Total working years')

    cmap = sns.cubehelix_palette(start=0.333333333333, light=1, as_

```

yaygın yoludur. İki değişken arasındaki ilişkinin gücünü ve yönünü ölçen -1 ile 1 arasında bir sayıdır.

Pearson korelasyon katsayısı da çıkarımsal bir istatistiktir, yani istatistiksel hipotezleri test etmek için kullanılabilir. Spesifik olarak, iki değişken arasında anlamlı bir ilişki olup olmadığını test edebiliriz.

- Pearson korelasyon katsayısı ne zaman kullanılır?

Pearson korelasyon katsayısı (r), bir korelasyonu ölçmek istediğinizde aralarından seçim yapmanız gereken birkaç korelasyon katsayısından biridir. Pearson korelasyon katsayısı, aşağıdakilerin tümü doğru olduğunda iyi bir seçimdir:

- Her iki değişken de nicel olduğunda.
- Değişkenlerden herhangi biri nitel ise farklı bir yöntem kullanmanız gerekecektir.
- Değişkenler normal olarak dağıtıldığında.
- Dağılımların yaklaşık olarak normal olup olmadığını doğrulamak için her bir değişkenin histogramını da oluşturabiliriz. Değişkenler biraz normal değilse sorun değil.
- Verilerde aykırı değer olmadığında.
- Aykırı değerler, verilerin geri kalanıyla aynı kalıpları takip etmeyen gözlemlerdir.
- İlişki doğrusal olduğunda.
- “Doğrusal”, iki değişken arasındaki ilişkinin düz bir çizgi ile makul ölçüde iyi tanımlanabileceği anlamına gelir.

```

cmap=True)
x = attrition['Age'].values
y = attrition['DailyRate'].values
sns.kdeplot(x, y, cmap=cmap, shade=True, ax=axes[0, 1])
axes[0, 1].set(title='Age against Daily Rate')

cmap = sns.cubehelix_palette(start=0.666666666667, light=1, as_
cmap=True)
x = attrition['YearsInCurrentRole'].values
y = attrition['Age'].values
sns.kdeplot(x, y, cmap=cmap, shade=True, ax=axes[0, 2])
axes[0, 2].set(title='Years in role against Age')

cmap = sns.cubehelix_palette(start=1.0, light=1, as_cmap=True)
x = attrition['DailyRate'].values
y = attrition['DistanceFromHome'].values
sns.kdeplot(x, y, cmap=cmap, shade=True, ax=axes[1, 0])
axes[1, 0].set(title='Daily Rate against DistanceFromHome')

cmap = sns.cubehelix_palette(start=1.333333333333, light=1, as_
cmap=True)
x = attrition['DailyRate'].values
y = attrition['JobSatisfaction'].values
sns.kdeplot(x, y, cmap=cmap, shade=True, ax=axes[1, 1])
axes[1, 1].set(title='Daily Rate against Job satisfaction')

cmap = sns.cubehelix_palette(start=1.666666666667, light=1, as_
cmap=True)
x = attrition['YearsAtCompany'].values
y = attrition['JobSatisfaction'].values
sns.kdeplot(x, y, cmap=cmap, shade=True, ax=axes[1, 2])
axes[1, 2].set(title='Daily Rate against distance')

cmap = sns.cubehelix_palette(start=2.0, light=1, as_cmap=True)
x = attrition['YearsAtCompany'].values
y = attrition['DailyRate'].values
sns.kdeplot(x, y, cmap=cmap, shade=True, ax=axes[2, 0])
axes[2, 0].set(title='Years at company against Daily Rate')

cmap = sns.cubehelix_palette(start=2.333333333333, light=1, as_
cmap=True)
x = attrition['RelationshipSatisfaction'].values
y = attrition['YearsWithCurrManager'].values
sns.kdeplot(x, y, cmap=cmap, shade=True, ax=axes[2, 1])
axes[2, 1].set(title='Relationship Satisfaction vs years with m
anager')

cmap = sns.cubehelix_palette(start=2.666666666667, light=1, as_
cmap=True)
x = attrition['WorkLifeBalance'].values
y = attrition['JobSatisfaction'].values
sns.kdeplot(x, y, cmap=cmap, shade=True, ax=axes[2, 2])
axes[2, 2].set(title='WorklifeBalance against Satisfaction')

f.tight_layout()

```

```

if __name__ == '__main__':

    attrition = readData()

    analyseData(attrition)

    pearsonCorr(attrition)

    attrition = attrition.drop(['Attrition_numerical'], axis=1)

    categorical = []

    for col, value in attrition.iteritems():
        if value.dtype == 'object':
            categorical.append(col)

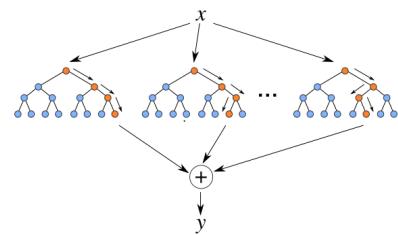
```

- Pearson korelasyon katsayısının hesaplanması

$$r = \frac{n \sum xy - (\sum x)(\sum y)}{\sqrt{[n \sum x^2 - (\sum x)^2][n \sum y^2 - (\sum y)^2]}}$$

Random Forrest

Random Forest: Birden fazla karar ağacının kullanılarak daha uyumlu modeller üreterek daha stabil bir sınıflandırma yapmya çalışan classification yapısıdır.



- Scikitlearn kütüphanesi içerisinde RandomForestClassifier çağrıldı. Bu fonksiyon ile tahmin verisi ile test tahmin verisi arasında karşılaştırma gerçekleştirmektedir.
- Regresyon sorunlarını çözmek için RFC kullanılırken verilerinizin her

```

numerical = attrition.columns.difference(categorical)

atr_cat = attrition[categorical]
atr_cat = atr_cat.drop(['Attrition'], axis=1)
atr_cat = pd.get_dummies(atr_cat)
atr_cat.head(3)
atr_num = attrition[numerical]
atr_fin = pd.concat([atr_num, atr_cat], axis=1)

target_map = {'Yes': 1, 'No': 0}

target = attrition["Attrition"].apply(lambda x: target_map[x])

train, test, target_train, target_val = train_test_split(atr_fin,
                                                         target,
                                                         train_size=0.80,
                                                         random_state=0);

oversampler = SMOTE(random_state=0)
smote_train, smote_target = oversampler.fit_resample(train, target_train)

seed = 0 # We set our random seed to zero for reproducibility
# Random Forest parameters
rf_params = {
    'n_jobs': -1,
    'n_estimators': 1000,
    # 'warm_start': True,
    'max_features': 0.3,
    'max_depth': 4,
    'min_samples_leaf': 2,
    'max_features': 'sqrt',
    'random_state': seed,
    'verbose': 0
}

rf = RandomForestClassifier(**rf_params)
rf.fit(smote_train, smote_target)
rf_predictions = rf.predict(test)
print("Accuracy score: {}".format(accuracy_score(target_val, rf_predictions)))
print("=" * 80)
print(classification_report(target_val, rf_predictions))

trace = go.Scatter(
    y=rf.feature_importances_,
    x=atr_fin.columns.values,
    mode='markers',
    marker=dict(
        sizemode='diameter',
        sizeref=1,
        size=13,
        # size= rf.feature_importances_,
        # color = np.random.randn(500), #set color equal to a variable
        color=rf.feature_importances_,
        colorscale='Portland',
        showscale=True
    ),
    text=atr_fin.columns.values
)
data = [trace]

layout = go.Layout(
    autosize=True,
    title='Random Forest Feature Importance',
    hovermode='closest',
    xaxis=dict(
        ticklen=5,
        showgrid=False,
        zeroline=False,

```

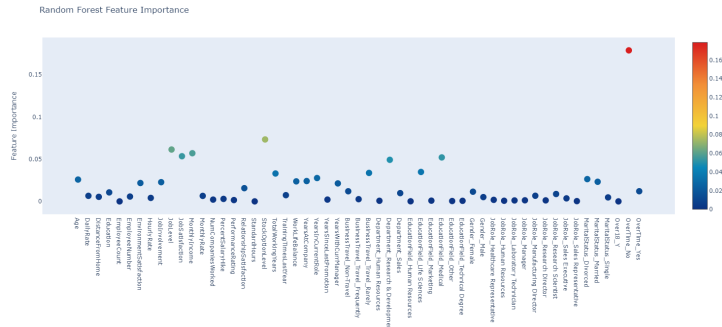
düğümünden nasıl dallanacağını gösteren ortalama kare hatası uygulanmaktadır.

- RFC algoritması ister regresyon ister sınıflandırma verileri olsun tüm farklı veri kümeleri türleri için son derece yararlı olabilir. Kullanımı kolaydır, eğitilmesi hızlıdır ve kullandığı karar ağaçlarının doğru bir temsili bulur.

```

        showline=False
    ),
    yaxis=dict(
        title='Feature Importance',
        showgrid=False,
        zeroline=False,
        ticklen=5,
        gridwidth=2
    ),
    showlegend=False
)
fig = go.Figure(data=data, layout=layout)
py.plot(fig, filename='scatter')

```



Acuraccy

```

rf = RandomForestClassifier(**rf_params)
rf.fit(smote_train, smote_target)
rf_predictions = rf.predict(test)
print("Accuracy score: {}".format(accuracy_score(target_val, rf_predictions)))
print("=" * 80)
print(classification_report(target_val, rf_predictions))

```

Accuracy score: 0.8537414965986394

	precision	recall	f1-score	support
0	0.90	0.93	0.91	245
1	0.57	0.49	0.53	49
accuracy			0.85	294
macro avg	0.74	0.71	0.72	294
weighted avg	0.85	0.85	0.85	294

Accuracy Score:

$$Accuracy = \frac{True_{positive} + True_{negative}}{True_{positive} + True_{negative} + False_{positive} + False_{negative}}$$


- Modelin tüm sınıflarda nasıl performans gösterdiğini tanımlayan bir ölçümdür. Sınıflar eşit öneme sahip kabul görür

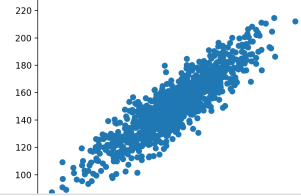
DoğruTahmin/ToplamTahmin

Kaynakça

How to Calculate Correlation Between Variables in Python - Machine Learning Mastery


There may be complex and unknown relationships between the variables in your dataset. It is important to discover and quantify the degree to which variables in your dataset are dependent upon each other.

 <https://machinelearningmastery.com/how-to-use-correlation-to-understand-the-relationship-between-variables/>



sklearn.ensemble.RandomForestClassifier


A random forest classifier. A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled

 <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>



What is Random Forest?


Learn about the random forest algorithm and how it can help you make better decisions to reach your business goals.

 <https://www.ibm.com/cloud/learn/random-forest>



IBM HR Analytics Employee Attrition & Performance

Predict attrition of your valuable employees

 <https://www.kaggle.com/datasets/pavansubhasht/ibm-hr-analytics-attrition-dataset>

