|  |  |
| --- | --- |
| **[1]** | **# Basic Library**  **import pandas as pd**  **import numpy as np**  **# Data Visualization**  **import matplotlib.pyplot as plt**  **from mpl\_toolkits.mplot3d import Axes3D**  **import seaborn as sns**  **from scipy.stats import skew**  **# Model Building**  **from sklearn.linear\_model import LinearRegression**  **import statsmodels.api as sm** |
| Line 1 adalah mengimport library yang dibutuhkan |
| **[2]** | **df\_train = pd.read\_csv('E:\Data\_Urang\Mata Kuliah\Semester 3\PROJECT II\Project New\PROJECT-2\dataset\employee\_attrition\_train.csv')**  **df\_train** |
| Line 2 mengimport data ke pyhton kemudian disimpan dalam variabel dengan nama df\_train. |
| **[3]** | **df\_train.info()** |
| Line 3 menampilkan info detail tabel/data yang disimpan. |
| **[4]** | **# Encoder BusinessTravel Variable**  **# converting type of columns to 'category'**  **df\_train['BusinessTravel']= df\_train['BusinessTravel'].astype('category')**  **# Assigning numerical values and storing in another column**  **df\_train['BusinessTravel']= df\_train['BusinessTravel'].cat.codes**  **# Encoder Department Variable**  **df\_train['Department']= df\_train['Department'].astype('category')**  **# Assigning numerical values and storing in another column**  **df\_train['Department'] = df\_train['Department'].cat.codes**  **# Encoder EducationField Variable**  **df\_train['EducationField']= df\_train['EducationField'].astype('category')**  **# Assigning numerical values and storing in another column**  **df\_train['EducationField']= df\_train['EducationField'].cat.codes**  **# Encoder Gender Variable**  **df\_train['Gender'] = df\_train['Gender'].astype('category')**  **# Assigning numerical values and storing in another column**  **df\_train['Gender'] = df\_train['Gender'].cat.codes**  **# Encoder JobRole Variable**  **df\_train['JobRole'] = df\_train['JobRole'].astype('category')**  **# Assigning numerical values and storing in another column**  **df\_train['JobRole'] = df\_train['JobRole'].cat.codes**  **# Encoder MaritalStatus Variable**  **df\_train['MaritalStatus']= df\_train['MaritalStatus'].astype('category')**  **# Assigning numerical values and storing in another column**  **df\_train['MaritalStatus'] = df\_train['MaritalStatus'].cat.codes**  **# Encoder Over18 Variable**  **df\_train['Over18'] = df\_train['Over18'].astype('category')**  **# Assigning numerical values and storing in another column**  **df\_train['Over18'] = df\_train['Over18'].cat.codes**  **# Encoder OverTime Variable**  **df\_train['OverTime'] = df\_train['OverTime'].astype('category')**  **# Assigning numerical values and storing in another column**  **df\_train['OverTime'] = df\_train['OverTime'].cat.codes**  **df\_train** |
| Line 4 melakukan encoder, karena machine learning tidak bisa membaca korelasi dari suatu variabel jika nilainya bernilai object/string. |
| **[5]** | **#Cek Apakah Ada Data yang Kosong?**  **df\_train.isnull().values.any()** |
| Line 5 cek apakah ada data yang kosong pada tabel. |
| **[6]** | **#Data kosong pada kolom**  **#Data yang kosong akan diolah untuk diisi dengan mean-nya**  **df\_train.isnull().sum()** |
| Line 6 cek berapa record yang terdapat data yag kosong. |
| **[7]** | **#Mengambil kolom Age untuk diisi dengan mean-nya**  **Age = df\_train['Age']**  **Age.describe()** |
| Line 7 membuat variabel Age yang berisikan atribut age, kemudian cek deskripsi variabel Age. |
| **[8]** | **df\_train.Age = df\_train.Age.fillna(value=df\_train.Age.mean())** |
| Line 8 melakukan pengisian data kosong pada variabel Age dengan nilai mean-nya. |
| **[9]** | **DailyRate = df\_train['DailyRate']**  **DailyRate.describe()** |
| Line 9 membuat variabel DailyRate yang berisikan atribut DailyRate, kemudian cek deskripsi variabel DailyRate. |
| **[10]** | **df\_train.DailyRate= df\_train.DailyRate.fillna**  **(value=df\_train.DailyRate.mean())** |
| Line 10 melakukan pengisian data kosong pada variabel DailyRate dengan nilai mean-nya. |
| **[11]** | **DistanceFromHome = df\_train['DistanceFromHome']**  **DistanceFromHome.describe()** |
| Line 11 membuat variabel DistanceFromHome yang berisikan atribut DistanceFromHome, kemudian cek deskripsi variabel DistanceFromHome. |
| **[12]** | **df\_train.DistanceFromHome= df\_train.DistanceFromHome.fillna**  **(value=df\_train.DistanceFromHome.mean())** |
| Line 12 melakukan pengisian data kosong pada variabel DistanceFromHome dengan nilai mean-nya. |
| **[13]** | **df\_train\_clean = df\_train**  **df\_train\_clean.isnull().values.any()** |
| Line 13 membuat variabel df\_train\_clean yang berisikan data df\_train, kemudian cek apakah masih terdapat data yang kosong atau tidak. |
| **[14]** | **df\_train\_clean.isnull().sum()** |
| Line 15 cek berapa record yang terdapat data yag kosong. |
| **[15]** | **def plotCorrelationMatrix(df\_train\_clean, graphWidth):**  **df\_train\_clean = df\_train\_clean[[col for col in df\_train\_clean if df\_train\_clean[col].nunique() > 1]] # keep columns where there are more than 1 unique values**  **if df\_train\_clean.shape[1] < 2:**  **print(f'No correlation plots shown: The number of non-NaN or constant columns ({df\_train\_clean.shape[1]}) is less than 2')**  **return**  **corr = df\_train\_clean.corr()**  **plt.figure(num=None, figsize=(graphWidth, graphWidth), dpi=80, facecolor='w', edgecolor='k')**  **corrMat = plt.matshow(corr, fignum = 1)**  **plt.xticks(range(len(corr.columns)), corr.columns, rotation=90)**  **plt.yticks(range(len(corr.columns)), corr.columns)**  **plt.gca().xaxis.tick\_bottom()**  **plt.colorbar(corrMat)**  **plt.title(f'Correlation Matrix for Data Training', fontsize=15)**  **plt.show()**  **plotCorrelationMatrix(df\_train\_clean, 8)** |
| Line 15 membuat heatmap antar variabel yang terdapat dalam variabel df\_train\_clean. |
| **[16]** | **df\_train\_clean.corr().abs()** |
| Line 16 membuat tabel korelasi antar variabel. |
| **[17]** | **df\_train\_clean.columns** |
| Line 17 menampilkan kumpulan variabel yang terdapat dalam df\_train\_clean. |
| **[18]** | **df\_train\_clean = df\_train\_clean.drop(['Attrition', 'BusinessTravel', 'DailyRate', 'Department','DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount','EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate', 'JobInvolvement', 'JobRole', 'JobSatisfaction', 'MaritalStatus','MonthlyRate', 'NumCompaniesWorked','Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating', 'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel', 'TrainingTimesLastYear', 'WorkLifeBalance','YearsInCurrentRole', 'YearsSinceLastPromotion',**  **'YearsWithCurrManager'], axis=1)** |
| Line 18 melakukan drop variabel yang tidak diperlukan. |
| **[19]** | **df\_train\_clean.corr()** |
| Line 19 membuat tabel korelasi setelah proses drop variabel. |
| **[20]** | **# Menentukan variabel X dan variabel Y**  **x\_train = df\_train\_clean[['Age', 'JobLevel', 'TotalWorkingYears', 'YearsAtCompany']]**  **y\_train = df\_train\_clean[['MonthlyIncome']]** |
| Line 20 menetapkan variabel independent (sumbu x) yaitu variabel Age, JobLevel, TotalWorkingYears, dan YearsAtCompany. Kemudian menetapkan variabel variabel dependen (sumbu y) yaitu MonthlyIncome. |
| **[21]** | **df\_train\_clean.to\_csv('E:\Data\_Urang\Mata Kuliah\Semester 3\PROJECT II\Project New\PROJECT-2\dataset\employee\_attrition\_train\_clean.csv')** |
| Line 21 melakukan export data ke format file csv. |
| **[22]** | **%matplotlib inline**  **plt.style.use('ggplot')**  **plt.rcParams['figure.figsize'] = (12,8)** |
| Line 22 membuat style untuk diagram plot. |
| **[23]** | **vis\_train = pd.read\_csv('E:\Data\_Urang\Mata Kuliah\Semester 3\PROJECT II\Project New\PROJECT-2\dataset\employee\_attrition\_train\_clean.csv')**  **vis\_train = vis\_train.drop(['Unnamed: 0'], axis=1)**  **sns.pairplot(vis\_train, x\_vars = ['Age', 'JobLevel', 'TotalWorkingYears', 'YearsAtCompany'], y\_vars = 'MonthlyIncome', height=5, aspect=0.7)** |
| Line 23 melakukan import data kemudian disimpan dalam variabel vis\_train. Drop kolom yang tidak diperlukan. Kemudian menentukan variabel x dan y untuk diagram plot. |
| **[24]** | **sns.heatmap(vis\_train.corr(), annot=True)** |
| Line 24 membuat heatmap untuk melihat korelasi antar variabel. |
| **[25]** | **df\_test = pd.read\_csv('E:\Data\_Urang\Mata Kuliah\Semester3\PROJECT II\Project New\PROJECT-2\dataset\employee\_attrition\_test.csv')**  **df\_test** |
| Line 25 melakukan import data testing kemudian disimpan dalam variabel df\_test. |
| **[26]** | **# Encoder BusinessTravel Variable**  **# converting type of columns to 'category'**  **df\_test['BusinessTravel'] = df\_test['BusinessTravel'].astype('category')**  **# Assigning numerical values and storing in another column**  **df\_test['BusinessTravel'] = df\_test['BusinessTravel'].cat.codes**  **# Encoder Department Variable**  **df\_test['Department'] = df\_test['Department'].astype('category')**  **# Assigning numerical values and storing in another column**  **df\_test['Department'] = df\_test['Department'].cat.codes**  **# Encoder EducationField Variable**  **df\_test['EducationField'] = df\_test['EducationField'].astype('category')**  **# Assigning numerical values and storing in another column**  **df\_test['EducationField'] = df\_test['EducationField'].cat.codes**  **# Encoder Gender Variable**  **df\_test['Gender'] = df\_test['Gender'].astype('category')**  **# Assigning numerical values and storing in another column**  **df\_test['Gender'] = df\_test['Gender'].cat.codes**  **# Encoder JobRole Variable**  **df\_test['JobRole'] = df\_test['JobRole'].astype('category')**  **# Assigning numerical values and storing in another column**  **df\_test['JobRole'] = df\_test['JobRole'].cat.codes**  **# Encoder MaritalStatus Variable**  **df\_test['MaritalStatus'] = df\_test['MaritalStatus'].astype('category')**  **# Assigning numerical values and storing in another column**  **df\_test['MaritalStatus'] = df\_test['MaritalStatus'].cat.codes**  **# Encoder Over18 Variable**  **df\_test['Over18'] = df\_test['Over18'].astype('category')**  **# Assigning numerical values and storing in another column**  **df\_test['Over18'] = df\_test['Over18'].cat.codes**  **# Encoder OverTime Variable**  **df\_test['OverTime'] = df\_test['OverTime'].astype('category')**  **# Assigning numerical values and storing in another column**  **df\_test['OverTime'] = df\_test['OverTime'].cat.codes**  **df\_test** |
| Line 26 melakukan encoder, karena machine learning tidak bisa membaca korelasi dari suatu variabel jika nilainya bernilai object/string. |
| **[27]** | **df\_test.isnull().values.any()** |
| Line 27 cek apakah masih terdapat data yang kosong atau tidak. |
| **[28]** | **df\_test.isnull().sum()** |
| Line 28 cek apakah ada data yang kosong pada tabel. |
| **[29]** | **# Missing Value in Age**  **Age = df\_test['Age']**  **df\_test.Age = df\_test.Age.fillna(value=df\_test.Age.mean())**  **# Missing Value in DailyDate**  **DailyRate = df\_test['DailyRate']**  **df\_test.DailyRate = df\_test.DailyRate.fillna(value=df\_test.DailyRate.mean())**  **# Missing Value in DistanceFromHome**  **DistanceFromHome = df\_test['DistanceFromHome']**  **df\_test.DistanceFromHome = df\_test.DistanceFromHome.fillna(value=df\_test.DistanceFromHome.mean())**  **# Missing Value in BusinessTravel**  **BusinessTravel = df\_test['BusinessTravel']**  **df\_test.BusinessTravel = df\_test.BusinessTravel.fillna(value=df\_test.BusinessTravel.mean())**  **# Missing Value in MartialStatus**  **MaritalStatus = df\_test['MaritalStatus']**  **df\_test.MaritalStatus = df\_test.MaritalStatus.fillna(value=df\_test.MaritalStatus.mean())** |
| Line 29 membuat variabel kemudian melakukan pengisian data kosong pada variabel dengan nilai mean-nya. |
| **[30]** | **df\_test.isnull().values.any()** |
| Line 30 cek apakah masih terdapat data yang kosong atau tidak. |
| **[31]** | **df\_test.isnull().sum()** |
| Line 31 cek apakah ada data yang kosong pada tabel. |
| **[32]** | **# Eliminasi Variabel yang Tidak akan digunakan**  **df\_test\_clean = df\_test.drop(['BusinessTravel', 'DailyRate', 'Department','DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount','EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate','JobInvolvement', 'JobRole', 'JobSatisfaction', 'MaritalStatus', 'MonthlyRate', 'NumCompaniesWorked','Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating', 'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel', 'TrainingTimesLastYear', 'WorkLifeBalance','YearsInCurrentRole', 'YearsSinceLastPromotion','YearsWithCurrManager'], axis=1)** |
| Line 32 melakukan drop variabel yang tidak diperlukan. |
| **[33]** | **df\_test\_clean.corr().abs()** |
| Line 33 membuat tabel korelasi setelah proses drop variabel. |
| **[34]** | **x\_test = df\_test\_clean[['Age', 'JobLevel', 'TotalWorkingYears', 'YearsAtCompany']]**  **y\_test = df\_test\_clean[['MonthlyIncome']]** |
| Line 34 menetapkan variabel independent (sumbu x) yaitu variabel Age, JobLevel, TotalWorkingYears, dan YearsAtCompany. Kemudian menetapkan variabel variabel dependen (sumbu y) yaitu MonthlyIncome. |
| **[35]** | **df\_test\_clean.to\_csv('E:\Data\_Urang\Mata Kuliah\Semester 3\PROJECT II\Project New\PROJECT-2\dataset\employee\_attrition\_test\_clean.csv')** |
| Line 35 melakukan export data ke format file csv. |
| **[36]** | **vis\_test = pd.read\_csv('E:\Data\_Urang\Mata Kuliah\Semester 3\PROJECT II\Project New\PROJECT-2\dataset\employee\_attrition\_test\_clean.csv')**  **vis\_test = vis\_test.drop(['Unnamed: 0'], axis=1)**  **sns.pairplot(vis\_test, x\_vars = ['Age', 'JobLevel', 'TotalWorkingYears', 'YearsAtCompany'], y\_vars = 'MonthlyIncome', height=5, aspect=0.7)** |
| Line 36 melakukan import data kemudian disimpan dalam variabel vis\_train. Drop kolom yang tidak diperlukan. Kemudian menentukan variabel x dan y untuk diagram plot. |
| **[37]** | **sns.heatmap(vis\_test.corr(), annot=True)** |
| Line 37 membuat heatmap untuk melihat korelasi antar variabel. |
| **[38]** | **regressor = LinearRegression()**  **persamaan = regressor.fit(x\_train, y\_train)**  **print(regressor.coef\_)**  **print(regressor.intercept\_)** |
| membuat variabel regressor yang isinya metode LinearRegression, kemudian membuat variabel persamaan yang isinya terdapat method regressor.fit dengan parameternya x\_train dan y\_train. |
| **[39]** | **y\_pred = regressor.predict(x\_test)**  **print(y\_pred)** |
| Mencari konstanta/intercept menggunakan regressor, kemudian ditampilkan. |
| **[40]** | **vis\_test['MonthlyIncome Prediction'] = y\_pred.tolist()**  **vis\_test** |
| Melakukan prediksi data testing menggunkana model machine learning. Kemudian buat kolom baru yang Bernama MonthlyIncome Prediction yang berisikan nilai prediksi. |
| **[41]** | **vis\_test.to\_excel('E:\Data\_Urang\Mata Kuliah\Semester 3\PROJECT II\Project New\PROJECT-2\dataset\employee\_vis\_test.xlsx')** |
| Melakukan export data setelah prediksi ke dalam format file excel. |
| **[42]** | **from sklearn.metrics import r2\_score**  **r2 = r2\_score (y\_test, y\_pred)**  **print ("Nilai R2 adalah ", r2)** |
| Dari scikit learn metrics diimport metode r2\_score. Kemudian hitung r2\_score dengan parameternya x\_train dan y\_train, kemudian hasilnya disimpan pada variable r2. Hasil perhitungan nilai pada variabel r2 ditampilkan. |
| **[43]** | **X = df\_train\_clean[['Age', 'JobLevel', 'TotalWorkingYears', 'YearsAtCompany']]**  **X = sm.add\_constant(X) # adding a constant**  **olsmod = sm.OLS(df\_train['MonthlyIncome'], X).fit()**  **print(olsmod.summary())** |
|  |
| **[44]** | **print('F-statistic:', olsmod.fvalue)**  **print('Probability of observing value at least as high as F-statistic:', olsmod.f\_pvalue)** |
|  |
| **[45]** | **print(olsmod.pvalues)** |
|  |
| **[46]** | **df\_test\_new = pd.read\_excel('E:\Data\_Urang\Mata Kuliah\Semester 3\PROJECT II\Project New\PROJECT-2\dataset\employee\_vis\_test.xlsx')**  **df\_test\_new2 = df\_test\_new.drop(['Unnamed: 0'], axis=1)**  **df\_test\_new2['MonthlyIncome Prediction'] = olsmod.predict(X)**  **df\_test\_new2['residual'] = olsmod.resid**  **df\_test\_new2** |
|  |
| **[47]** | **# Plotting the observed vs predicted values**  **sns.lmplot(x='MonthlyIncome', y='MonthlyIncome Prediction', data=df\_test\_new, fit\_reg=False, size=5)**  **# Plotting the diagonal line**  **line\_coords = np.arange(vis\_test[['MonthlyIncome', 'MonthlyIncome Prediction']].min().min()-10,**  **vis\_test[['MonthlyIncome', 'MonthlyIncome Prediction']].max().max()+10)**  **plt.plot(line\_coords, line\_coords, # X and y points**  **color='darkorange', linestyle='--')**  **plt.ylabel('Predicted MonthlyIncome', fontsize=14)**  **plt.xlabel('Actual MonthlyIncome', fontsize=14)**  **plt.title('Linearity Assumption', fontsize=16)**  **plt.show()** |
|  |
| [48] | from statsmodels.stats.diagnostic import normal\_ad  # Performing the test on the residuals  p\_value = normal\_ad(df\_test\_new2['residual'])[1]  print('p-value from the test Anderson-Darling test below 0.05 generally means non-normal:', p\_value)  # Plotting the residuals distribution  plt.subplots(figsize=(8, 4))  plt.title('Distribution of Residuals', fontsize=18)  sns.distplot(df\_test\_new2['residual'])  plt.show()  # Reporting the normality of the residuals  if p\_value < 0.05:  print('Residuals are not normally distributed')  else:  print('Residuals are normally distributed') |
|  |
| [49] | corr = vis\_test[['Age', 'JobLevel', 'TotalWorkingYears', 'YearsAtCompany', 'MonthlyIncome']].corr()  print('Pearson correlation coefficient matrix of each variables:\n', corr)  # Generate a mask for the diagonal cell  mask = np.zeros\_like(corr, dtype=np.bool)  np.fill\_diagonal(mask, val=True)  # Initialize matplotlib figure  fig, ax = plt.subplots(figsize=(4, 3))  # Generate a custom diverging colormap  cmap = sns.diverging\_palette(220, 10, as\_cmap=True, sep=100)  cmap.set\_bad('grey')  # Draw the heatmap with the mask and correct aspect ratio  sns.heatmap(corr, mask=mask, cmap=cmap, vmin=-1, vmax=1, center=0, linewidths=.5)  fig.suptitle('Pearson correlation coefficient matrix', fontsize=14)  ax.tick\_params(axis='both', which='major', labelsize=10)  # fig.tight\_layout() |
|  |
| [50] | from statsmodels.stats.stattools import durbin\_watson  durbinWatson = durbin\_watson(df\_test\_new2['residual'])  print('Durbin-Watson:', durbinWatson)  if durbinWatson < 1.5:  print('Signs of positive autocorrelation', '\n')  print('Assumption not satisfied')  elif durbinWatson > 2.5:  print('Signs of negative autocorrelation', '\n')  print('Assumption not satisfied')  else:  print('Little to no autocorrelation', '\n')  print('Assumption satisfied') |
|  |
| [51] | # Plotting the residuals  plt.subplots(figsize=(8, 4))  plt.scatter(x=df\_test\_new2.index, y=df\_test\_new2.residual, alpha=0.8)  plt.plot(np.repeat(0, len(df\_test\_new2.index)+2), color='darkorange', linestyle='--')  plt.ylabel('Residual', fontsize=14)  plt.xlabel('Week', fontsize=14)  plt.title('Homescedasticity Assumption', fontsize=16)  plt.show() |
|  |
| [52] | # Urutan Inputan : Age, JobLevel (1-5), TotalWorkingYears, YearsAtCompany  salary\_pred = regressor.predict([[20, 1, 3, 1]])  print("Gaji yang terprediksi pada pegawai perbulan yang bekerja sepanjang tahun tersebut adalah ",salary\_pred) |
|  |
| [53] | import pickle  pd.to\_pickle(persamaan, "model3.pickle") |
|  |