# BAB IV

# IMPLEMENTASI DAN PENGUJIAN

Setelah data dan sistem dianalisis dan didesain secara rinci, maka selanjutnya adalah tahapan implementasi. Implementasi dapat diartikan sebagai proses melaksanakan penerapan dari ide atau konsep yang telah disusun. Dengan begitu, implementasi tidak hanya sekedar melakukan penerapan suatu konsep, tetapi juga merupakan suatu rencana kegiatan yang telah tersusun untuk mencapai tujuan kegiatan. Pada tahap implementasi ini mencakup uji coba sistem dan penerapan antarmuka yang telah dirancang sebelumnya.

Perancangan sistem yang telah dibuat akan dilakukan penerapannya ke dalam bentuk web base. Dengan menggunakan *software multiplatform*, sistem dapat dirancang sesuai dengan perencanaan yang telah disusun. Perancangan ini meliputi penerapan antarmuka sistem ke dalam bentuk web base dan pengujian pada sistem yang telah dibuat. Pengujian sistem bertujuan untuk mengetahui tingkat keberhasilan berjalannya suatu sistem yang dibuat.

Tahap implementasi adalah tahapan yang penting dalam pembuatan sebuah sistem. Dengan adanya tahap ini, dapat diketahui suatu kendala dan seperti apa sistem yang telah dirancang tersebut dapat berjalan. Sehingga, jika ditemukannya suatu kendala pada sistem, sistem yang dibuat dapat diperbaiki kembali sesuai tujuan yang akan dicapai.

## **Implementasi Antarmuka (*Interface*)**

Impelementasi sistem memiliki bagian berupa pemaparan mengenai tampilan pada sistem serta kegunaan dari setiap halaman. Implementasi dari hasil perancanggan menggunakan bahasa pemrograman python dengan framework Django.

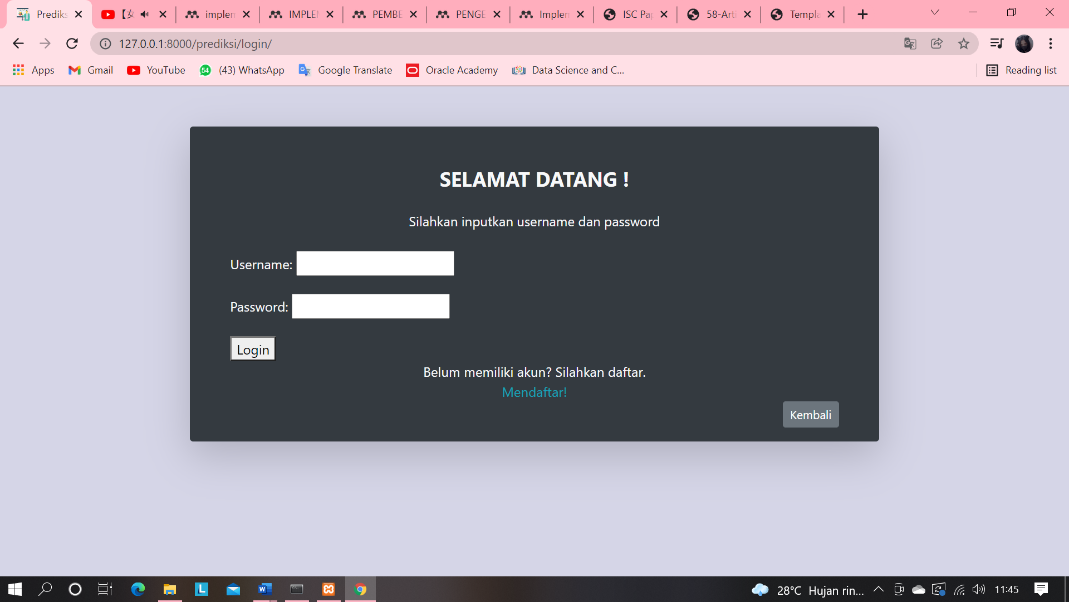
### **Implementasi Halaman Depan**



Gambar 1 Halaman Depan Sistem

Gambar diatas merupakan implementasi dari halaman depan sistem. Pada halaman ini, *user* dapat menekan button “Login” pada navbar untuk masuk ke dalam sistem.

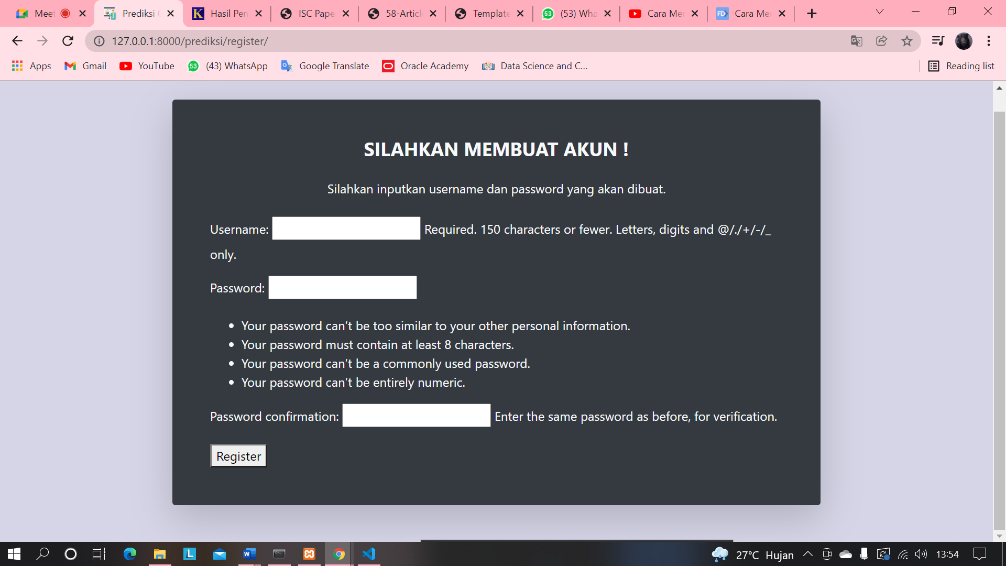
### **Implementasi Halaman *Login***



Gambar 2 Halaman Login Sistem

Gambar diatas merupakan implementasi dari halaman *login*. Sebelum masuk ke halaman *dashboard*, *user* harus menginputkan *username* dan *password* yang sesuai. Jika sesuai, maka *user* akan di arahkan ke halaman *dashboard*. Jika tidak *user* akan diminta kembali memasukkan *username* dan *password* yang sesuai.

### **Implementasi Halaman Registrasi**



Gambar 3 Halaman Registrasi Sistem

Gambar diatas merupakan implementasi dari halaman registrasi. Jika *user* belum memiliki akun untuk masuk sistem, maka *user* dapat melakukan registrasi dengan cara menginputkan *username* dan *password* lalu menekan tombol register.

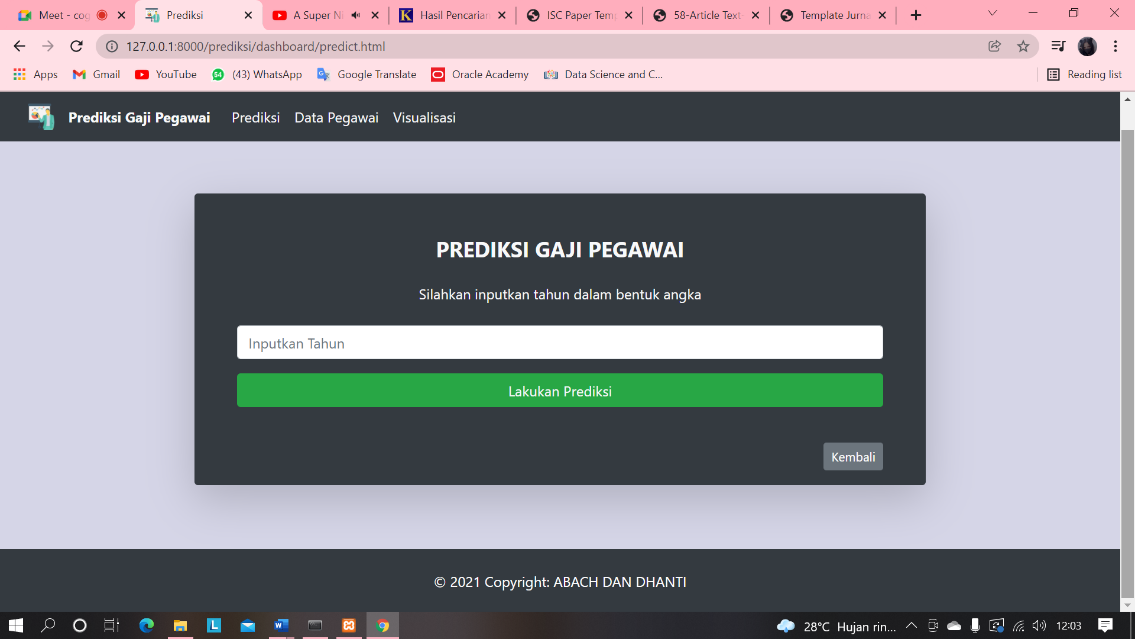
### **Implementasi Halaman *Dashboard***



Gambar 4 Halaman Dashboard Sistem

Gambar diatas merupakan implementasi dari halaman *dashboard* dari sistem yang dirancang. Pada bagian navbar, terdapat pilihan submenu untuk mengakses suatu halaman yang lainnya. Pilihan submenu tersebut terdiri dari Prediksi, Data Pegawai, serta Visualisasi.

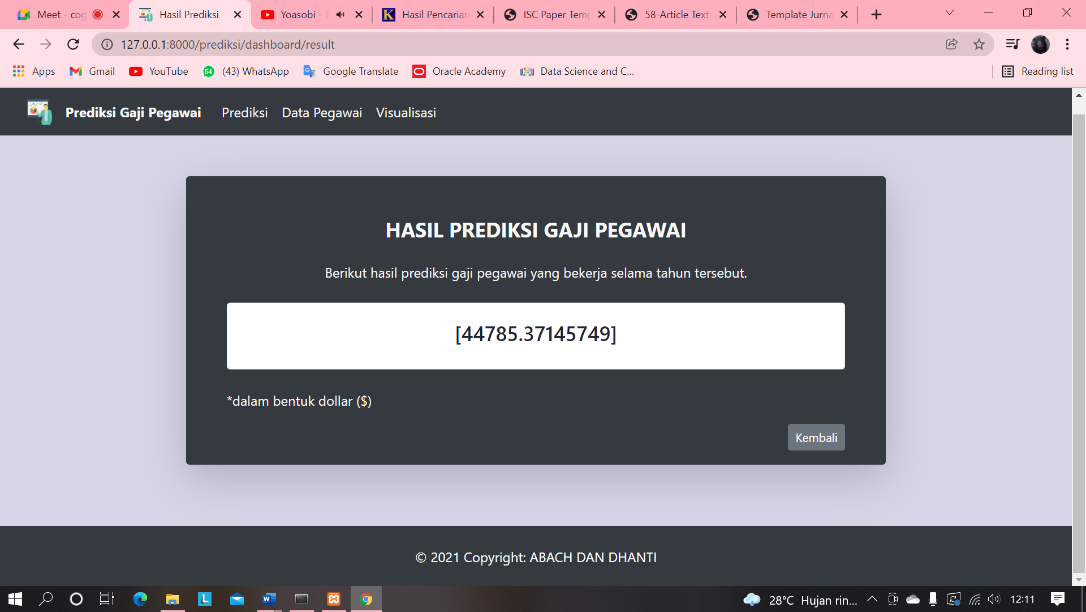
### **Implementasi Halaman Prediksi**



Gambar 5 Halaman Prediksi Sistem

Gambar diatas merupakan implementasi dari halaman prediksi dari sistem yang dirancang. Pada halaman ini, terdapat form untuk melakukan prediksi gaji pegawai. *User* dapat menginputkan angka ke dalam form untuk mengetahui hasil prediksi gaji pegawai. Kemudian *user* dapat menekan button “Lakukan Prediksi” untuk melihat hasil prediksi berdasarkan tahun yang diinputkan.

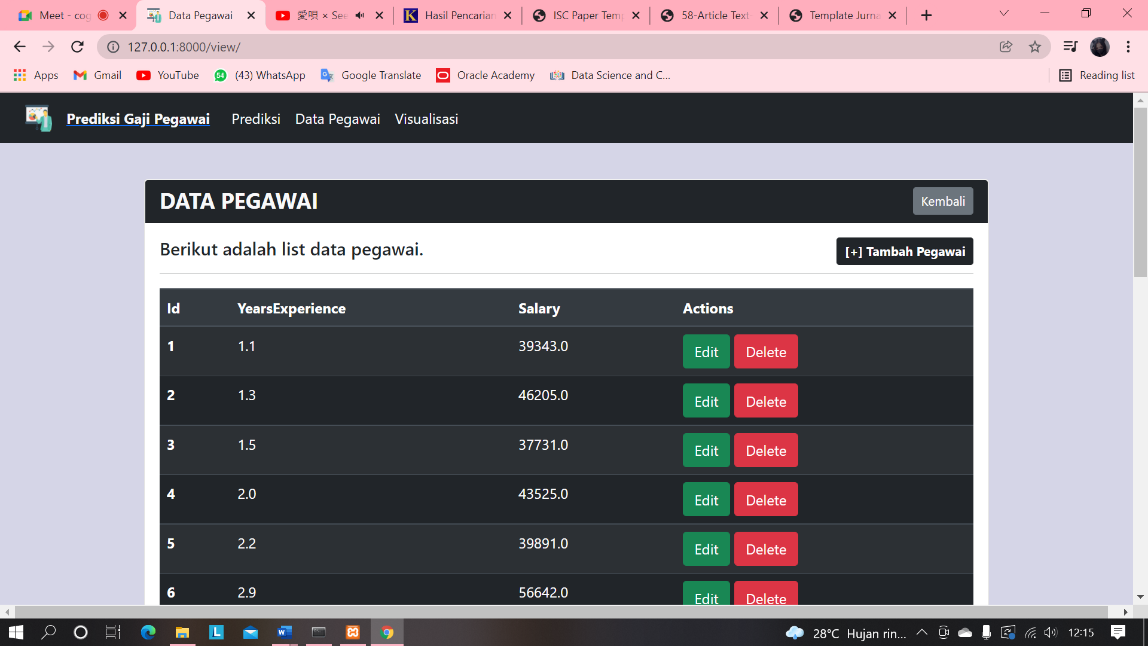
### **Implementasi Halaman Hasil Prediksi**



Gambar 6 Halaman Hasil Prediksi Sistem

Gambar diatas merupakan implementasi dari halaman hasil prediksi dari sistem yang dirancang. Pada halaman ini, akan ditampilkan hasil prediksi gaji dari user yang sebelumnya telah menginputkan angka tahun pada form prediksi.

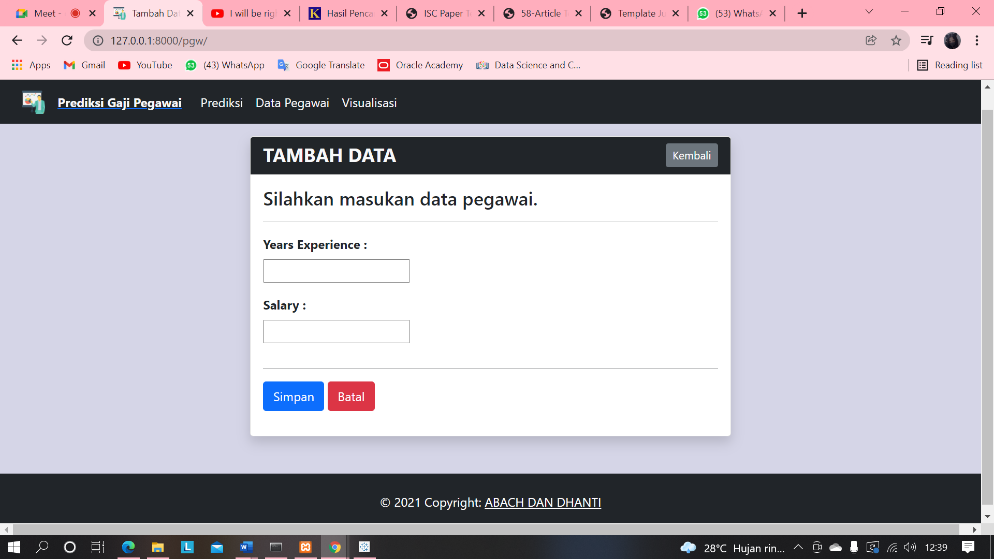
### **Implementasi Halaman Data Pegawai**



Gambar 7 Halaman Data Pegawai Sistem

Gambar diatas merupakan implementasi dari halaman data pegawai dari sistem yang dirancang. Pada halaman ini, akan ditampilkan data dalam bentuk tabel yang terdiri dari atribut Id, YearsExperience, Salary, dan Action. *Record* yang ditampilkan terdiri dari 22 *record*. Terdapat *button* Tambah Pegawai yang berfungsi untuk menambah data pegawai, *button* Edit berfungsi untuk mengubah suatu *record* yang dipilih, dan *button* Delete untuk menghapus suatu *record* yang dipilih.

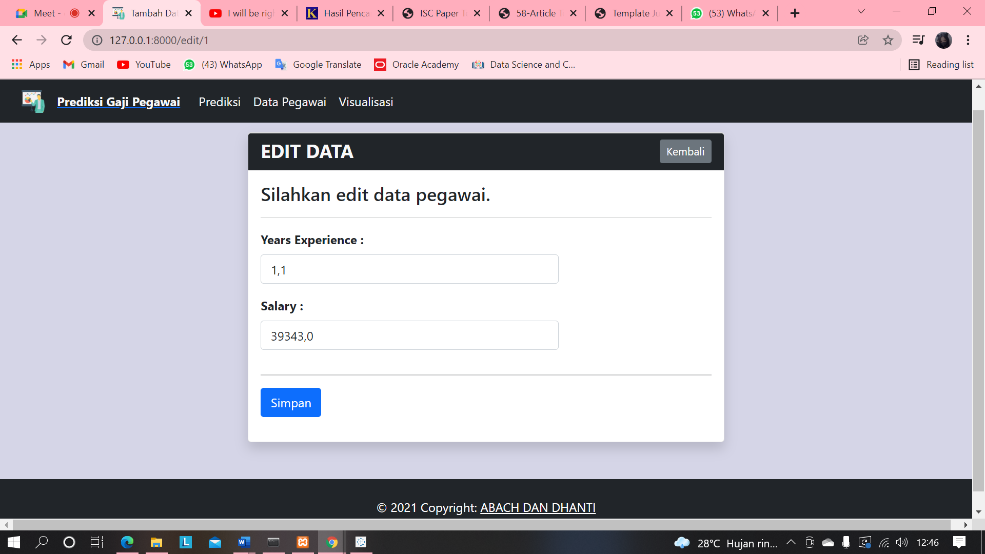
### **Implementasi Halaman Tambah Data Pegawai**



Gambar 8 Halaman Tambah Data Pegawai Sistem

Gambar diatas merupakan implementasi dari halaman tambah data pegawai dari sistem yang dirancang. Pada halaman ini, akan ditampilkan form tambah data untuk menambah data pegawai. *User* dapat menginputkan Years Experience dan Salary pada form. Lalu *user* dapat menekan *button* Simpan untuk menyimpan data yang diinputkan. *User* dapat menekan *button* Batal untuk me-*reset* data yang diinputkan pada form.

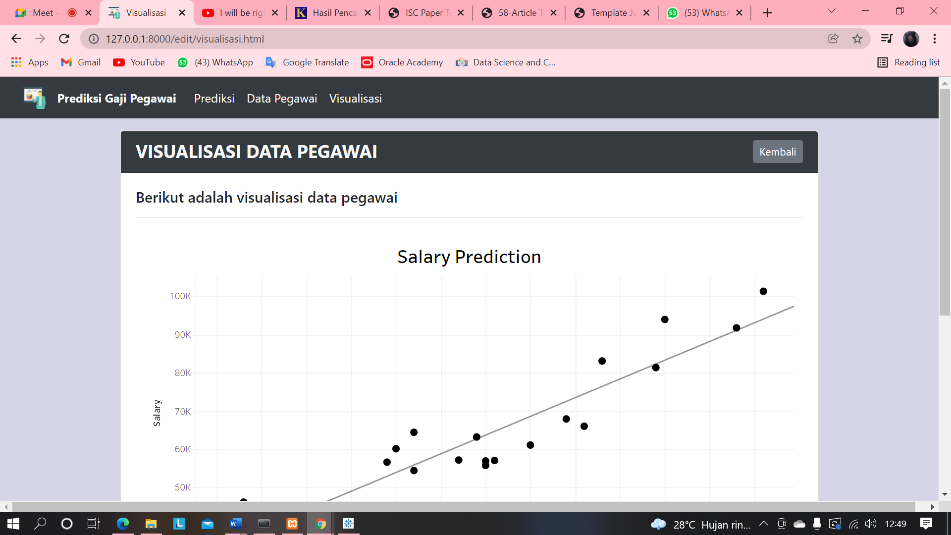
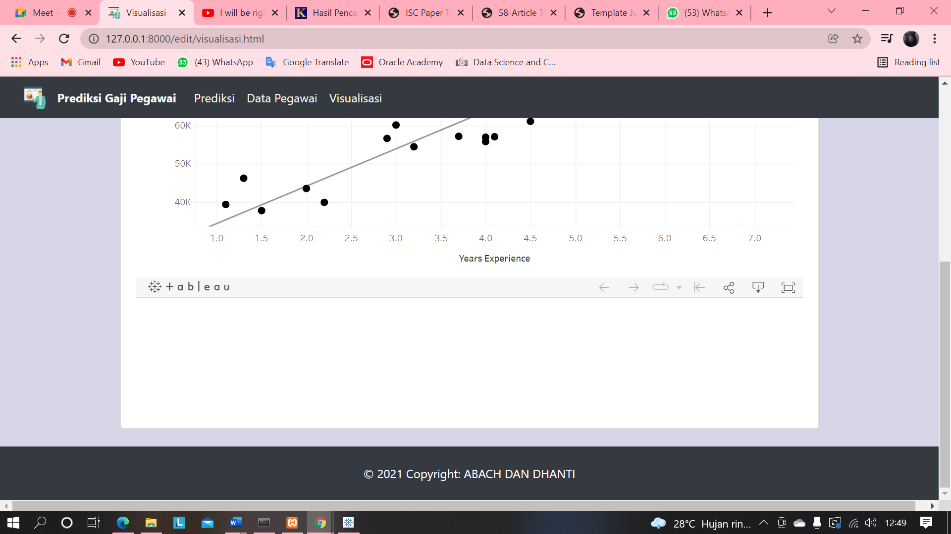
### **Implementasi Halaman Edit Data Pegawai**



Gambar 9 Halaman Edit Data Pegawai Sistem

Gambar diatas merupakan implementasi dari halaman edit data pegawai dari sistem yang dirancang. Pada halaman ini, akan ditampilkan form edit data untuk mengubah data pegawai. *User* dapat menginputkan Years Experience dan Salary pada form. Lalu *user* dapat menekan *button* Simpan untuk menyimpan data yang telah diubah.

### **Implementasi Halaman Visualisasi**



Gambar 10 Halaman Visualisasi Sistem

Gambar diatas merupakan implementasi dari halaman visualisasi dari sistem yang dirancang. Pada halaman ini, akan ditampilkan bentuk visualisasi prediksi antara gaji (*salary*) dengan pengalaman lama bekerja (*Years Experience*).

## **Implementasi Aplikasi**

Pada bagian implementasi aplikasi, dipaparkan kode-kode pada setiap proses berjalannya sistem yang telah dibangun. Pada bagian ini, pemaparan pola desain arsitektur dalam sistem. Arsitektur tersebut ialah Views dan Template serta URL.

### **Implementasi Proses Login**

1. Views

@login\_required

def dashboardView(request):

    return render(request, 'dashboard.html')

1. Template

<h1 class="h4 text-light mb-4">

    <b>SELAMAT DATANG !</b>

    </h1>

    <div class="sidebar-brand-text mx-3 text-light">

    Silahkan inputkan username dan password

       </div>

       <br/>

       </div>

       <form class="text-light" method="POST">

          {% csrf\_token %} {{form.as\_p}}

       <button>Login</button>

       <br/>

       <div align="center"class="sidebar-brand-text mx-3 text-light">

          Belum memiliki akun? Silahkan daftar.

       </div>

       <div align="center">

       <a href="http://127.0.0.1:8000/prediksi/register/"class="text info"> Mendaftar!</a>

       </div>

       <div class="float-right">

       <a href=<http://127.0.0.1:8000> class="btn btn-secondary btn-sm">Kembali</a>

       </div>

       </form>

1. URL

 path('login/',LoginView.as\_view(),name="login\_url"),

### **Implementasi Proses Registrasi**

1. Views

def registerView(request):

    if request.method == "POST":

        form = UserCreationForm(request.POST)

        if form.is\_valid():

            form.save()

            return redirect('login\_url')

    else:

        form = UserCreationForm()

    return render(request, 'registration/register.html', {'form': form})

1. Template

  <b>SILAHKAN MEMBUAT AKUN !</b>

       </h1>

    <div class="sidebar-brand-text mx-3 text-light">

    Silahkan inputkan username dan password yang akan dibuat.

    </div>

    <br/>

      </div>

      <form method="POST" class="text-light">

        {% csrf\_token %} {{form.as\_p}}

      <button>Register</button>

      </form>

1. URL

 path('register/', views.registerView, name="register\_url"),

### **Implementasi Proses Prediksi Gaji**

1. Views

def predict(request):

    context={'a':1}

    return render(request, 'predict.html', context)

1. Template

<h1 class="h4 text-light mb-4"><b>PREDIKSI GAJI PEGAWAI</b></h1>

<div class="sidebar-brand-text mx-3 text-light"> Silahkan inputkan tahun dalam bentuk angka </div>

    <br>

     </div>

     {% block content %}

     <form class="user" method="POST" action="result">

        {% csrf\_token %}

       <div class="form-group">

       <input type="float" name="tahun" id="tahun" class="form-control"  placeholder="Inputkan Tahun" autocomplete="off" required >

       </div>

       <div class="form-row">

        <div class="form-group col-md-12">

       <input type="submit" value="Lakukan Prediksi" class="btn btn-block bg-success text-light">

        </div>

       </div>

         {% endblock %}

      </form>

         {{ result }}

1. URL

url('predict', views.predict, name='predict')

### **Implementasi Proses Hasil Prediksi Gaji**

1. Views

def result(request):

    print(request)

    tahun = float(request.POST.get('tahun'))

    model = pd.read\_pickle('./models/model3.pickle')

    result = model.predict([[tahun]])

    return render(request, 'result.html', {'result': result})

1. Template

 <h1 class="h4 text-light mb-4"><b> HASIL PREDIKSI GAJI PEGAWAI</b></h1>

     <div class="sidebar-brand-text mx-3 text-light"> Berikut hasil prediksi gaji pegawai yang bekerja selama tahun tersebut.</div>

     <br>

     </div>

     <div class="card text-center">

         <div class="card-body">

             <h4>{{ result }}</h4>

         </div>

     </div>

     <br>

     <p class="text-light">\*dalam bentuk dollar ($)</p>

1. URL

url('result', views.result, name='result')

### **Implementasi Proses Menampilkan Data Pegawai**

1. Views

def view(request):

    pegawai = Pegawai.objects.all()

    return render(request, "view.html", {'pegawai': pegawai})

1. Template

<h5>Berikut adalah list data pegawai. </h5>

    </div>

    <div class="float-right">

      <a href="http://127.0.0.1:8000/pgw" class="btn btn-dark btn-sm"><b>[+] Tambah Pegawai</b></a>

    </div>

    <br>

    <hr>

<table class="table table-dark table-striped">

  <thead class="thead-dark">

    <tr>

      <th scope="col">Id</th>

      <th scope="col">YearsExperience</th>

      <th scope="col">Salary</th>

      <th scope="col">Actions</th>

    </tr>

  </thead>

  {% for pegawai in pegawai %}

    <tr>

      <th scope="row">{{ pegawai.id }}</th>

      <td>{{ pegawai.YearsExperience }}</td>

      <td>{{ pegawai.Salary }}</td>

      <td>

          <a href="/edit/{{ pegawai.id }}"><span class="btn btn-success">Edit</span></a>

          <a href="/delete/{{ pegawai.id }}"><span class="btn btn-danger">Delete</span></a>

      </td>

    </tr>

  {% endfor %}

  </tbody>

</table>

1. URL

path('view/', views.view),

### **Implementasi Proses Tambah Data Pegawai**

1. Views

def pgw(request):

    if request.method == "POST":

        form = PegawaiForm(request.POST)

        if form.is\_valid():

            try:

                form.save()

                return redirect('/view')

            except:

                pass

    else:

        form = PegawaiForm()

    return render(request, 'haltambah.html', {'form': form})

1. Template

<h4>Silahkan masukan data pegawai.</h4>

    <hr>

    <form action="/pgw/" id="form-tambah" method="POST">

      {% csrf\_token %}

    <div class="form-row">

       <div class="form-group col-md-8">

           <label for="kode\_produk"><strong>Years Experience :</strong></label>

           <div>{{ form.YearsExperience }}</div>

       </div>

    </div>

    <div class="form-row">

        <div class="form-group col-md-8">

            <label for="jenis\_produk"><strong>Salary : </strong></label>

            <div>{{ form.Salary }}</div>

        </div>

    </div>

    <hr>

    <div class="form-group">

       <button type="submit" class="btn btn-primary">Simpan</button>

       <button type="reset" class="btn btn-danger">Batal</button>

    </div>

    </form>

1. URL

 path('pgw/', views.pgw),

### **Implementasi Proses Edit Data Pegawai**

1. Views

def edit(request, id):

    pegawai = Pegawai.objects.get(id=id)

    return render(request, 'edit.html', {'pegawai': pegawai})

def update(request, id):

    pegawai = Pegawai.objects.get(id=id)

    form = PegawaiForm(instance=pegawai)

    if request.method == 'POST':

        form = PegawaiForm(request.POST, instance=pegawai)

        if form.is\_valid():

            form.save()

            return redirect('/view')

    return render(request, 'view.html', {'form': form})

1. Template

 <h4>Silahkan edit data pegawai.</h4>

   <hr>

  <form action="/update/{{pegawai.id}}" class="post-form" method="POST">

  <input type="hidden" name="id" id="id" required maxlength="20" value="{{ pegawai.id }}"/>

      {% csrf\_token %}

   <div class="form-row">

       <div class="form-group col-md-8">

        <label for="kode\_produk"><strong>Years Experience :</strong></label>

   <div>

       <input type="number" class="form-control" name="YearsExperience" value="{{ pegawai.YearsExperience }}">

             </div>

          </div>

      </div>

      <div class="form-row">

         <div class="form-group col-md-8">

             <label for="jenis\_produk"><strong>Salary : </strong></label>

          <div>

             <input  type="number" class="form-control"  name="Salary" value="{{ pegawai.Salary }}">

            </div>

         </div>

      </div>

      <hr>

      <div class="form-group">

         <button type="submit" class="btn btn-primary">Simpan</button>

      </div>

</form>

1. URL

path('edit/<int:id>', views.edit),

path('update/<int:id>', views.update),

### **Implementasi Proses Hapus Data Pegawai**

1. Views

def delete(request, id):

    pegawai = Pegawai.objects.get(id=id)

    pegawai.delete()

    return redirect("/view")

1. URL

path('delete/<int:id>', views.delete),

### **Implementasi Proses Menampilkan Visualisasi**

1. Views

def visualisasi(request):

    return render(request, 'visualisasi.html')

1. Template

<div class="float-left">

  <h5>Berikut adalah visualisasi data pegawai </h5>

</div>

  <br>

  <hr>

  <div class='tableauPlaceholder' id='viz1640335357904' style='position: relative'>

    <object class='tableauViz'  style='display:none;'>

    <param name='host\_url' value='https%3A%2F%2Fpublic.tableau.com%2F'/>

    <param name='embed\_code\_version' value='3' />

    <param name='site\_root' value='' />

    <param name='name' value='SalaryVSYearsExperience4&#47;Dashboard8'/>

    <param name='tabs' value='no' />

    <param name='toolbar' value='yes' />

    <param name='animate\_transition' value='yes' />

    <param name='display\_static\_image' value='yes' />

    <param name='display\_spinner' value='yes' />

    <param name='display\_overlay' value='yes' />

    <param name='display\_count' value='yes' />

    <param name='language' value='en-US' />

    <param name='filter' value='publish=yes' />

   </object>

</div>

  <script type='text/javascript'>

  var divElement = document.getElementById('viz1640335357904');

  var vizElement = divElement.getElementsByTagName('object')[0];

  if ( divElement.offsetWidth > 800 )

  {

    vizElement.style.minWidth='600px';

    vizElement.style.maxWidth='900px';

    vizElement.style.width='100%';

    vizElement.style.minHeight='127px';

    vizElement.style.maxHeight='527px';

    vizElement.style.height=(divElement.offsetWidth\*0.75)+'px';

    }

    else if ( divElement.offsetWidth > 500 )

    {

      vizElement.style.minWidth='600px';

      vizElement.style.maxWidth='900px';

      vizElement.style.width='100%';

      vizElement.style.minHeight='127px';

      vizElement.style.maxHeight='527px';

      vizElement.style.height=(divElement.offsetWidth\*0.75)+'px';

      }

      else

      {

        vizElement.style.width='100%';

        vizElement.style.height='727px';

        }

      var scriptElement = document.createElement('script');

        scriptElement.src = 'https://public.tableau.com/javascripts/api/viz\_v1.js';

        vizElement.parentNode.insertBefore(scriptElement, vizElement);

        </script>

      </div>

1. URL

 url('visualisasi', views.visualisasi, name='visualisasi'),

## **Implementasi Model *Machine Learning***

Pada bagian implementasi model *machine learning*, dipaparkan kode-kode dalam proses analisis data pegawai berdasarkan pengalaman lama bekerja. Selain analisis data, dilakukan juga pembuatan model prediksi gaji pegawai.

### **Himpunan Data**

Pada tahapan ini kita perlu untuk memahami dan mempersiapkan data yang dikenal dengan istilah *Data Preprocessing*. Metode yang digunakan dalam *Data Preprocessing* pada model ini adalah *Data Cleaning*. Berikut kode programnya :

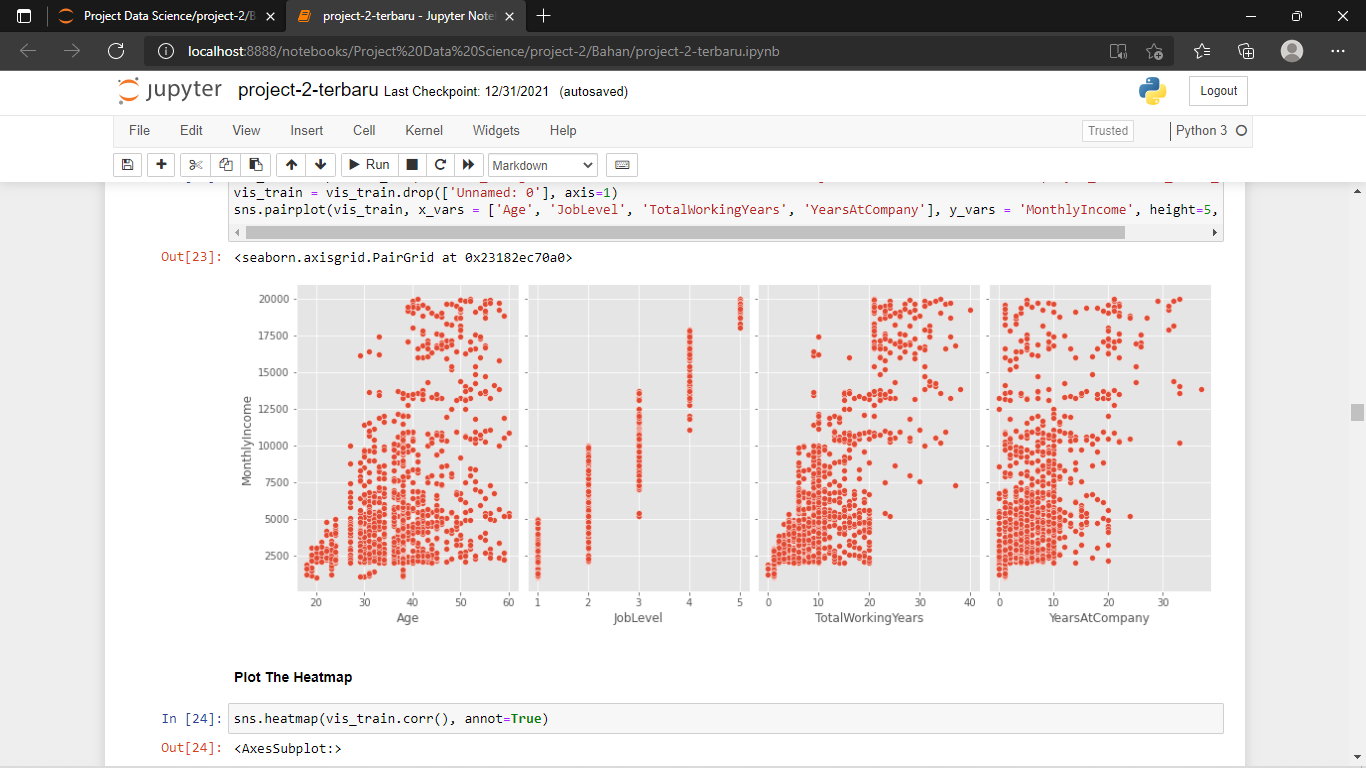
|  |  |
| --- | --- |
| [1] | # Basic Library  import pandas as pd  import numpy as np  # Data Visualization  import matplotlib.pyplot as plt  from mpl\_toolkits.mplot3d import Axes3D  import seaborn as sns  from scipy.stats import skew  # Model Building  from sklearn.linear\_model import LinearRegression  import statsmodels.api as sm |
| [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 |
| [3] | df\_train.info() |
| [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) |
| [5] | df\_train.isnull().values.any() |
| [6] | df\_train.isnull().sum() |
| [7] | Age = df\_train['Age']  Age.describe() |
| [8] | df\_train.Age = df\_train.Age.fillna(value=df\_train.Age.mean()) |
| [9] | DailyRate = df\_train['DailyRate']  DailyRate.describe() |
| [10] | df\_train.DailyRate= df\_train.DailyRate.fillna  (value=df\_train.DailyRate.mean()) |
| [11] | DistanceFromHome = df\_train['DistanceFromHome']  DistanceFromHome.describe() |
| [12] | df\_train.DistanceFromHome= df\_train.DistanceFromHome.fillna  (value=df\_train.DistanceFromHome.mean()) |
| [13] | df\_train\_clean = df\_train  df\_train\_clean.isnull().values.any() |
| [14] | df\_train\_clean.isnull().sum() |
| [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) |
| [16] | df\_train\_clean.corr().abs() |
| [17] | df\_train\_clean.columns |
| [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) |
| [19] | df\_train\_clean.corr() |
| [20] | x\_train = df\_train\_clean[['Age', 'JobLevel', 'TotalWorkingYears', 'YearsAtCompany']]  y\_train = df\_train\_clean[['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') |
| [22] | %matplotlib inline  plt.style.use('ggplot')  plt.rcParams['figure.figsize'] = (12,8) |
| [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) |
| [24] | sns.heatmap(vis\_train.corr(), annot=True) |

Keterangan :

* Line 1 : Mengimport library yang dibutuhkan
* Line 2 : Mengimport data ke pyhton kemudian disimpan dalam variabel dengan nama df\_train.
* Line 3 : Menampilkan info detail tabel/data yang disimpan.
* Line 4 : Melakukan encoder, karena machine learning tidak bisa membaca korelasi dari suatu variabel jika nilainya bernilai object/string.
* Line 5 : Cek apakah ada data yang kosong pada tabel.
* Line 6 : Cek berapa record yang terdapat data yag kosong.
* Line 7 : Membuat variabel Age yang berisikan atribut age, kemudian cek deskripsi variabel Age.
* Line 8 : Melakukan pengisian data kosong pada variabel Age dengan nilai mean-nya.
* Line 9 : Membuat variabel DailyRate yang berisikan atribut DailyRate, kemudian cek deskripsi variabel DailyRate.
* Line 10 : Melakukan pengisian data kosong pada variabel DailyRate dengan nilai mean-nya.
* Line 11 : Membuat variabel DistanceFromHome yang berisikan atribut DistanceFromHome, kemudian cek deskripsi variabel DistanceFromHome.
* Line 12 : Melakukan pengisian data kosong pada variabel DistanceFromHome dengan nilai mean-nya.
* Line 13 : Membuat variabel df\_train\_clean yang berisikan data df\_train, kemudian cek apakah masih terdapat data yang kosong atau tidak.
* Line 14 : Cek berapa record yang terdapat data yag kosong.
* Line 15 : Membuat heatmap antar variabel yang terdapat dalam variabel df\_train\_clean.
* Line 16 : Membuat tabel korelasi antar variabel.
* Line 17 : Menampilkan kumpulan variabel yang terdapat dalam df\_train\_clean.
* Line 18 : Melakukan drop variabel yang tidak diperlukan.
* Line 19 : Membuat tabel korelasi setelah proses drop variabel.
* Line 20 : Menetapkan variabel independent (sumbu x) yaitu variabel Age, JobLevel, TotalWorkingYears, dan YearsAtCompany. Kemudian menetapkan variabel variabel dependen (sumbu y) yaitu MonthlyIncome.
* Line 21 : Melakukan export data ke format file csv.

1. Visualisasi Data Training

|  |  |
| --- | --- |
| [22] | %matplotlib inline  plt.style.use('ggplot')  plt.rcParams['figure.figsize'] = (12,8) |
| [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) |
| [24] | sns.heatmap(vis\_train.corr(), annot=True) |



Gambar 11 Visualisasi Data Training

Keterangan :

* Line 22 : Membuat style untuk diagram plot.
* 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.
* Line 24 : Membuat heatmap untuk melihat korelasi antar variabel.

### **Proses Data Mining & Pengetahuan**

Pada tahapan Proses Data Mining kita perlu memilih metode yang sesuai dengan karakter data yang dikenal dengan istilah Modelling. Pada model ini digunakan Proses Data Mining Prediction. Pada tahapan Pengetahuan kita perlu memahami model dan pengetahuan yang sesuai sehingga dapat memilih model. Model yang digunakan adalah Linear Regression menggunakan Scikit Learn.

|  |  |
| --- | --- |
| [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 |
| [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 |
| [27] | df\_test.isnull().values.any() |
| 28] | df\_test.isnull().sum() |
| [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()) |
| [30] | df\_test.isnull().values.any() |
| [31] | df\_test.isnull().sum() |
| [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) |
| [33] | df\_test\_clean.corr().abs() |
| [34] | x\_test = df\_test\_clean[['Age', 'JobLevel', 'TotalWorkingYears', 'YearsAtCompany']]  y\_test = df\_test\_clean[['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') |
| [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) |
| [37] | sns.heatmap(vis\_test.corr(), annot=True) |
| [38] | regressor = LinearRegression()  persamaan = regressor.fit(x\_train, y\_train)  print(regressor.coef\_)  print(regressor.intercept\_) |
| [39] | y\_pred = regressor.predict(x\_test)  print(y\_pred) |
| [40] | vis\_test['MonthlyIncome Prediction'] = y\_pred.tolist()  vis\_test |
| [41] | vis\_test.to\_excel('E:\Data\_Urang\Mata Kuliah\Semester 3\PROJECT II\Project New\PROJECT-2\dataset\employee\_vis\_test.xlsx') |

Keterangan :

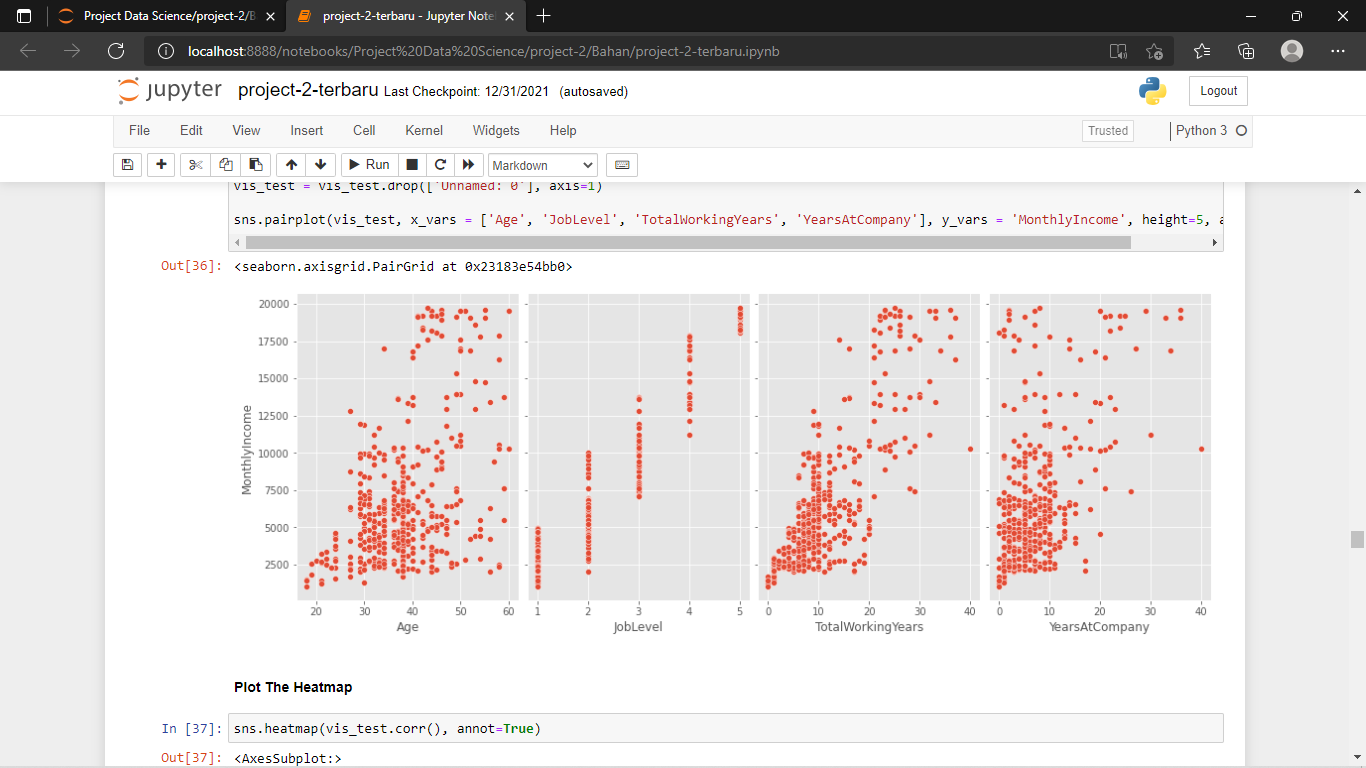
* Line 25 : Melakukan import data testing kemudian disimpan dalam variabel df\_test.
* Line 26 : Melakukan encoder, karena machine learning tidak bisa membaca korelasi dari suatu variabel jika nilainya bernilai object/string.
* Line 27 : Cek apakah masih terdapat data yang kosong atau tidak.
* Line 28 : Cek apakah ada data yang kosong pada tabel.
* Line 29 : Membuat variabel kemudian melakukan pengisian data kosong pada variabel dengan nilai mean-nya.
* Line 30 : Cek apakah masih terdapat data yang kosong atau tidak.
* Line 31 : Cek apakah ada data yang kosong pada tabel.
* Line 32 : Melakukan drop variabel yang tidak diperlukan.
* Line 33 : Membuat tabel korelasi setelah proses drop variabel.
* Line 34 : Menetapkan variabel independent (sumbu x) yaitu variabel Age, JobLevel, TotalWorkingYears, dan YearsAtCompany. Kemudian menetapkan variabel variabel dependen (sumbu y) yaitu MonthlyIncome.
* Line 35 : Melakukan export data ke format file csv.
* 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.
* Line 37 : membuat heatmap untuk melihat korelasi antar variabel.
* Line 38 : Membuat variabel regressor yang isinya metode LinearRegression, kemudian membuat variabel persamaan yang isinya terdapat method regressor.fit dengan parameternya x\_train dan y\_train.
* Pada bagian Age memiliki nilai koefisien -5,054, Job Level memiliki nilai koefisien 3871,75, TotalWorkingYears memiliki nilai koefisien 46,9405, dan YearsAtCompany memiliki nilai koefisien -9,8460. Nilai koefisien -5,054 tersebut memiliki arti bahwa tiap pertambahan satu umur karyawan, maka penurunan gaji sebesar 5,054. Kemudian, nilai koefisien 3871,75 memiliki arti bahwa tiap karyawan yang mengalami pertambahan satu level pekerjaan, akan diterima sebesar 3871,75. Nilai koefisien 46,9405 memiliki arti bahwa tiap karyawan yang mengalami pertambahan satu tahun pengalaman kerja, maka kenaikan gaji yang diterima adalah 46,94. Nilai koefisien -9,8460 tersebut memiliki arti bahwa tiap karyawan yang mengalami pertambahan satu tahun kerja, maka penurunan gaji yang diterima sebesesar 9,8460.
* Line 39 : Mencari konstanta/intercept menggunakan regressor, kemudian ditampilkan.
* Intecept merupakan sebuah koefisien dari sebuah persamaan model regresi linear. Untuk kasus ini, berarti untuk variabel X yang bernilai nol atau karyawan yang belum berpengalaman kerja, karyawan tersebut akan dikurangi gaji sebesar -1728,52 per tahunnya.
* Berdasarkan nilai koefisien Years Experience dan Intersept didapat persamaan regresi linear berikut :

**MonthlyIncome = -1728,52 - 5,054(Age) + 3871,75(JobLevel) + 46,9405(TotalWorkingYears) – 9,8460(YearsAtCompany)**

* Line 40 : Melakukan prediksi data testing menggunkana model machine learning. Kemudian buat kolom baru yang Bernama MonthlyIncome Prediction yang berisikan nilai prediksi.
* Line 41 : Melakukan export data setelah prediksi ke dalam format file excel.

1. Visualisasi Data Testing

|  |  |
| --- | --- |
| [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) |
| [37] | sns.heatmap(vis\_test.corr(), annot=True) |



Gambar 12 Visualisasi Data Testing

Keterangan :

* 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.
* Line 37 : Membuat heatmap untuk melihat korelasi antar variabel.

### **Evaluasi Data**

Pada tahap ini kita melakukan evaluasi terhadap analisis model dan kinerja metode.

1. R-Square

R2-Square adalah sebuah nilai yang menyatakan seberapa sesuai hasil prediksi model mendekati data yang sebenarnya. Semakin besar r2\_square, maka hasil prediksi semakin dekat dengan data yang sebenarnya, artinya sama saja dengan semakin mendekati 1, maka semakin bagus model tersebut.

|  |  |
| --- | --- |
| [42] | from sklearn.metrics import r2\_score  r2 = r2\_score (y\_test, y\_pred)  print ("Nilai R2 adalah ", r2) |

Keterangan :

* Line 42 : 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.

1. Validasi Model

|  |  |
| --- | --- |
| [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()) |

Keterangan :

* Line 43 : Melakukan penentuan nilai x sebagai nilai independent dan nilai y (olsmod) sebagai nilai dependen. Kemudian dengan model OLS dibuat kesimpulan validasi model yang memuat beberapa informasi seperti, R-Squared, F-Statistics, Prob (F-Statistic)/P-value, dan lain-lain.

1. Uji F (ANOVA)

F-test atau ANOVA (Analysis of variance) dalam regresi multi-linier dapat digunakan untuk menentukan apakah model kompleks kita berkinerja lebih baik daripada model yang lebih sederhana (misalnya model dengan hanya satu variabel independen). Dengan uji-F kita dapat mengevaluasi signifikansi model kita dengan menghitung probabilitas pengamatan statistik-F yang setidaknya setinggi nilai yang diperoleh model kita. Mirip dengan skor R², kita bisa dengan mudah mendapatkan statistik F dan probabilitas statistik F tersebut dengan mengakses atribut .fvaluesdan .f\_pvaluesdari model kita seperti di bawah ini.

|  |  |
| --- | --- |
| [44] | print('F-statistic:', olsmod.fvalue)  print('Probability of observing value at least as high as F-statistic:', olsmod.f\_pvalue) |

Keterangan :

* Line 43 : Menampilkan nilai Fs (fvalues) dan nilai P-value.
* Fs

Fs adalah hasil akhir dari analisis ANOVA. Nilai Fs ini yang dikenal dengan F hitung dalam pengujian hipotesa dibandingkan dengan nilai p-value. Jika Fs > P-value, maka dapat dinyatakan bahwa secara simultan (bersama-sama) Salary dan Years Experience berpengaruh signifikan terhadap permintaan.

* P-Value

P-Value adalah nilai probabilitas yang dapat diartikan sebagai besarnya peluang (probabilitas) yang diamati dari statistic uji. Variabel independent (Years Experience) mempunyai nilai p-value dibawah 0.00, hal ini menandakan ia signifikan melakukan prediksi variabel dependen (Salary).

Berdasarkan tabel diatas, dapat diketahui bahwa Fs > P-value, yang artinya variabel Salary dan Years Experience berpengaruh signifikan terhadap permintaan.

1. Uji-t

Statistik-t adalah koefisien dibagi dengan kesalahan standarnya. Kesalahan standar adalah perkiraan deviasi standar koefisien, jumlahnya bervariasi di setiap kasus. Ini dapat dianggap sebagai ukuran ketepatan yang digunakan untuk mengukur koefisien regresi. Sama seperti uji-F, nilai-p menunjukkan probabilitas untuk melihat hasil yang ekstrim seperti yang dimiliki model kita. Kita juga bisa mendapatkan nilai p untuk semua variabel kita dengan memanggil .pvalues atribut pada model.

|  |  |
| --- | --- |
| [45] | print(olsmod.pvalues) |

Keterangan :

* Line 45 : Menampilkan nilai p untuk semua variabel.

1. Pengujian Asumsi

Selanjutnya kita akan memvalidasi model dengan melakukan analisis residual.

|  |  |
| --- | --- |
| [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 |

Keterangan :

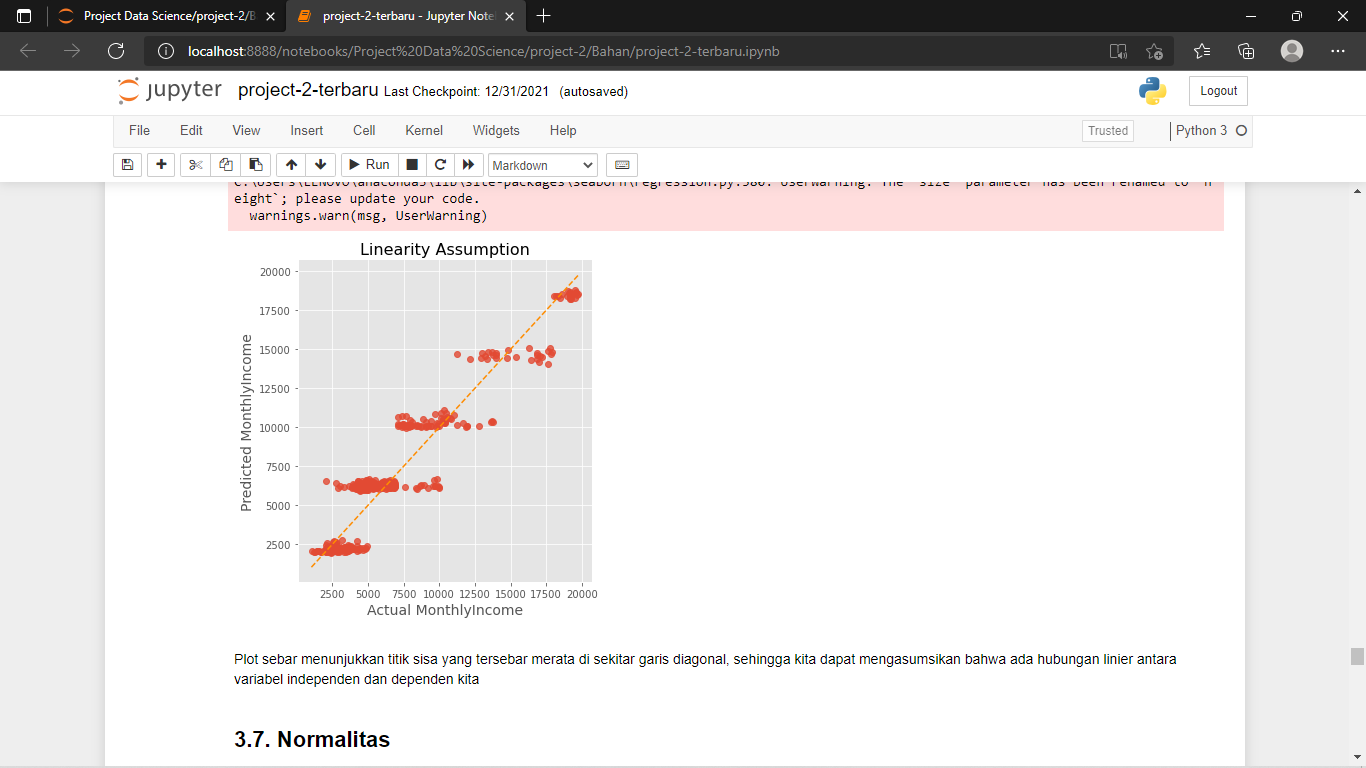
* Line 46 : Melakukan import data kemudian disimpan dalam variabel df\_test\_new. Kemudian lakukan drop kolom yang tidak bernilai. Setelah itu menentukan kolom MonthlyIncome sebagai kolom hasil prediksi dan kolom residual sebagai residual.

Berikut adalah daftar pengujian atau asumsi yang akan dilakukan untuk melihat validitas model :

1. Linearitas

Ini mengasumsikan bahwa terdapat hubungan linear antara variabel bebas dan variabel terikat. Dalam kasus ini, kami memiliki beberapa variabel independent, kami dapat melakukan ini dengan menggunakan plot pencar untuk melihat nilai prediksi kami dengan nilai sebenarnya.

|  |  |
| --- | --- |
| [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() |



Gambar 13 Linearitas Model

Keterangan :

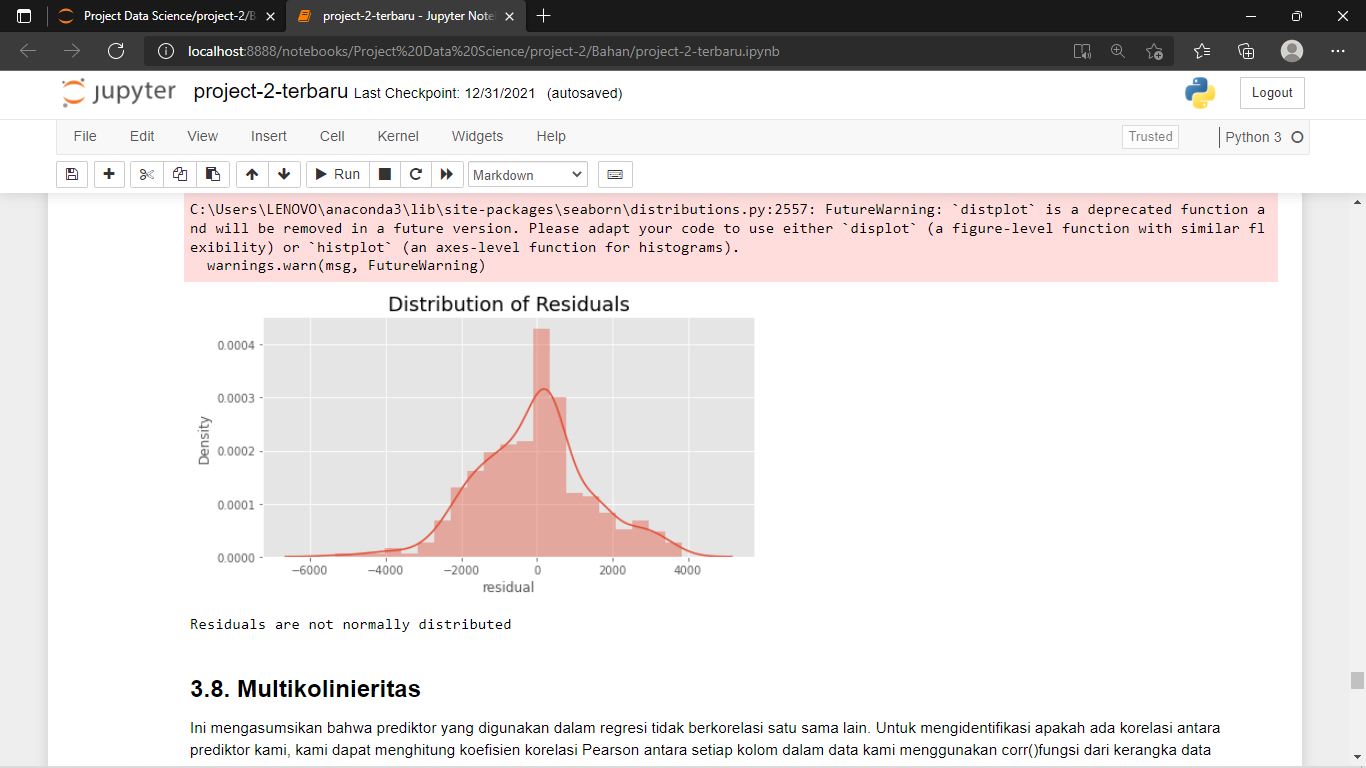
* Line 47 : Melakukan plot data dengan menentukan variabel x yaitu MonthlyIncome dengan variabel y yaitu MonthlyIncome Prediction. Setelah itu ditampilkan penyebaran datanya beserta garis linearnya.

Plot sebar menunjukkan titik sisa yang tersebar merata di sekitar garis diagonal, sehingga kita dapat mengasumsikan bahwa ada hubungan linear antara variabel independent dan dependen.

1. Normalitas

Ini mengasumsikan bahwa istilah kesalahan model terdistribusi normal. Kami akan memeriksa normalitas residual dengan memplotnya ke dalam histogram dan melihat nilai p dari uji normalitas Anderson-Darling. Kita akan menggunakan normal\_ad()fungsi dari statsmodel untuk menghitung p-value kita dan kemudian membandingkannya dengan threshold 0,05. Jika p-value yang kita peroleh lebih tinggi dari threshold, maka kita dapat mengasumsikan bahwa residual kita terdistribusi normal.

|  |  |
| --- | --- |
| [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') |



Gambar 14 Distribusi dari Nilai Residual

Keterangan :

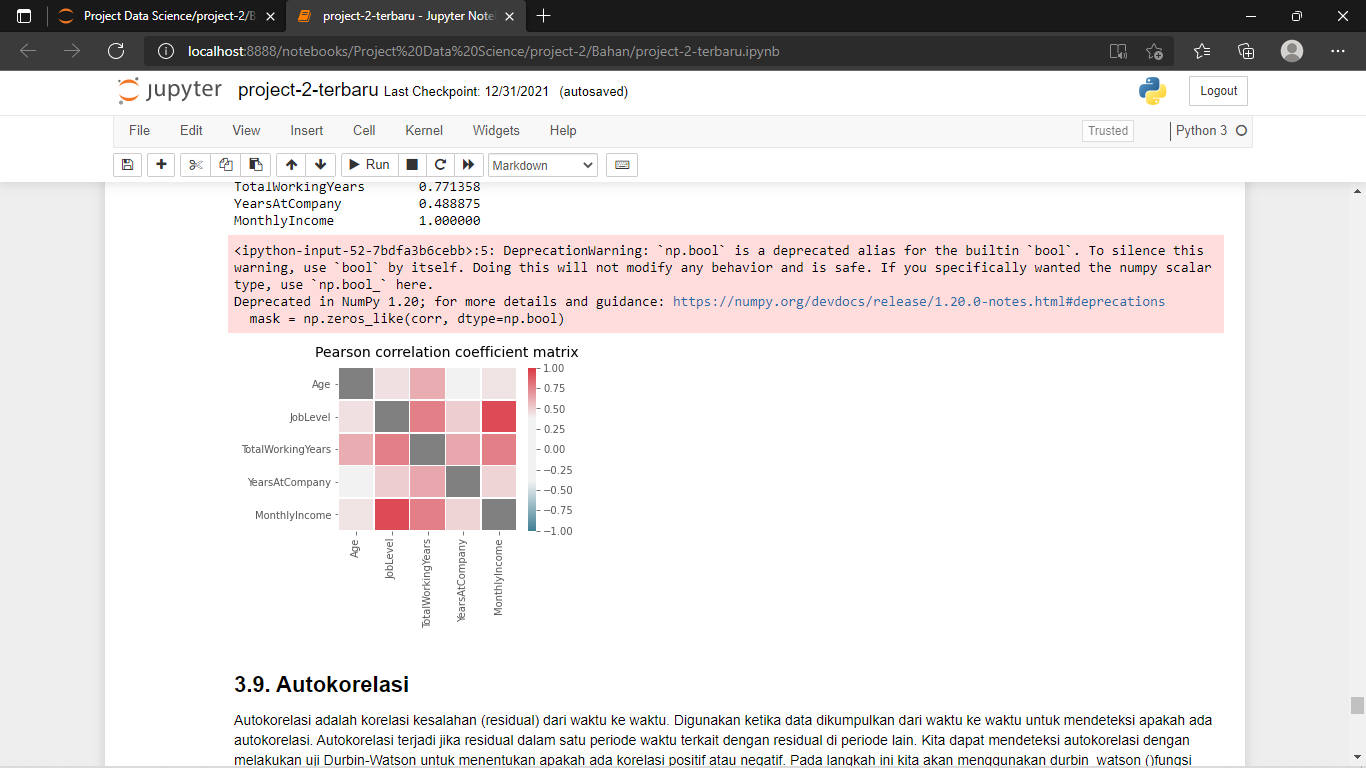
* Line 48 : Melakukan import library untuk melakukan normalitas. Kemudian, menentukan nilai threshold = 0,05 untuk dibandingkan dengan nilai p-value.

Didapat nilai p-value adalah 0,00032261 yang berarti nilai p-value kurang dari thresholdnya. Dari pernyataan tersebut dapat diambil kesimpulan bahwa Residual tidak terdistribusi secara normal.

1. Multikolinieritas

Ini mengasumsikan bahwa predictor yang digunakan dalam regresi berkorelasi satu sama lain. Untuk mengidentifikasi apakah ada korelasi antara variabel independent dengan dependen, dapat dihitung koefisien korelasi Pearson setiap kolom.

|  |  |
| --- | --- |
| [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() |



Gambar 15 Matriks Korelasi Pearson

Keterangan :

* Line 49 : Menampilkan matriks korelasi/heatmap dari variabel independent dan dependen.

1. Autokorelasi

Autokorelasi adalah korelasi kesalahan (residual) dari waktu ke waktu. Digunakan ketika data dikumpulkan dari waktu ke waktu untuk mendeteksi apakah ada autokoreasi. Autokorelasi terjadi jika residual dalam satu periode waktu terkait dengan residual di periode lain. Kita dapat mendeteksi autokorelasi dengan melakukan uji Durbin-Watson untuk menentukan apakah ada korelasi positif atau negatif. Pada langkah ini kita akan menentukan durbin\_watson( ) fungsi dari statsmodel untuk menghitung skor Durbin-Watson kita dan kemudian menilai nilainya dengan kondisi berikut.

1. Jika skor Durbin-Watson kurang dari 1,5 maka terdapat autokorelasi positif dan asumsi tidak terpenuhi.
2. Jika skor Durbin-Watson antara 1,5 dan 2,5 maka tidak ada autokorelasi dan asumsi puas.
3. Jika skor Durbin-Watson lebih dari 2,5 maka terdapat autokorelasi negatif dan asumsi tidak puas.

|  |  |
| --- | --- |
| [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') |

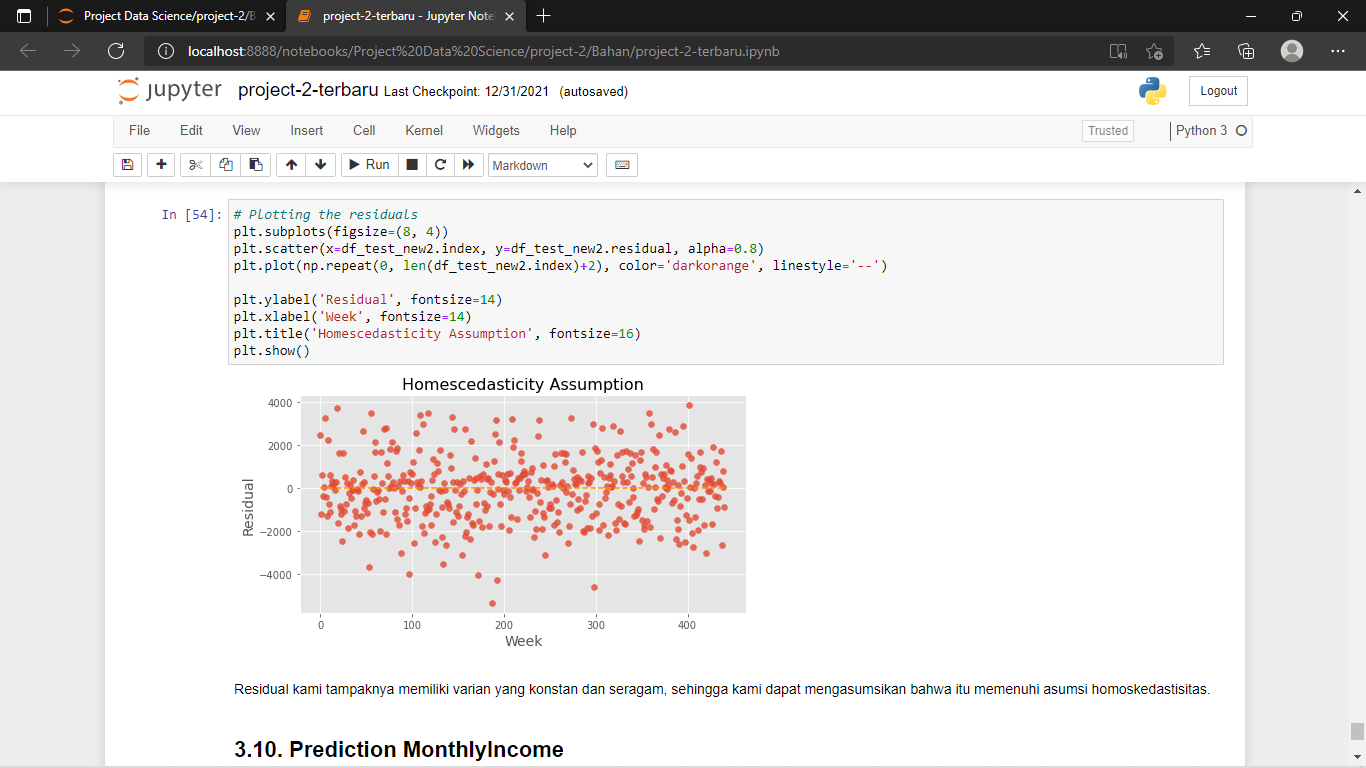
Keterangan :

* Line 50 : Melakukan import library durbin\_watson untuk mencari aurokorelasi pada data. Setelah itu, dilakukan penentuan autokorelasi dari nilai residual.

1. Homoskedastisitas

Melihat asumsi diatas, ini mengasumsikan homskedastisitas yang merupakan varian yang saam dalam istilah kesalahan kami. Heteroskedastisitas, pelanggaran homoskedastisitas, terjadi ketika kita tidak memiliki varian genap di selutuh istilah kesalahan. Untuk mendeteksi homoskedastisitas, kita dapat memplot residual kita dan melihat apakah variansnya tampak seragam.

|  |  |
| --- | --- |
| [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() |



Gambar 16 Asumsi Homoskedastisitas

Keterangan :

* Line 51 : Untuk membuat penyebaran varian data pada grafik asumsi homoskedastisitas.

Residual kami tampaknya memiliki varian yang konstan dan seragam, sehingga kami dapat mengasumsikan bahwa itu memenuhi asumsi homoskedastisitas.

1. Prediksi Gaji

|  |  |
| --- | --- |
| [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) |

Keterangan :

* Line 52 : Menentukan prediksi gaji dengan parameter Age = 20 tahun, JobLevel = 1, TotalWorkingYears = 3 tahun, dan YearsAtCompany = 1 tahun menggunakan metode pada variabel regressor, kemudian hasilnya ditampilkan.