# 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.

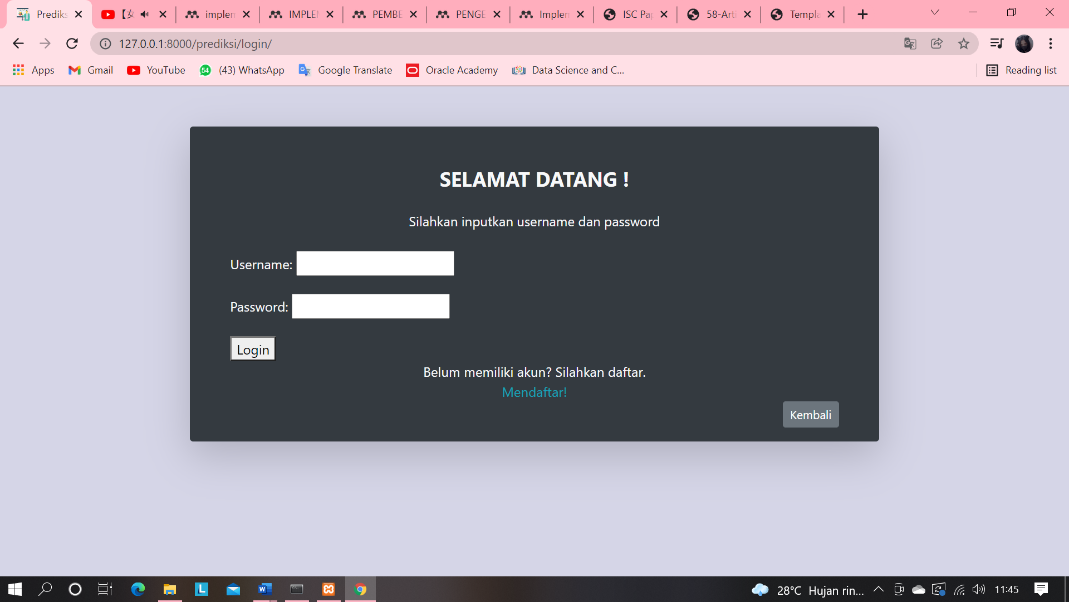
### **Implementasi Halaman Depan**



Gambar 21 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.

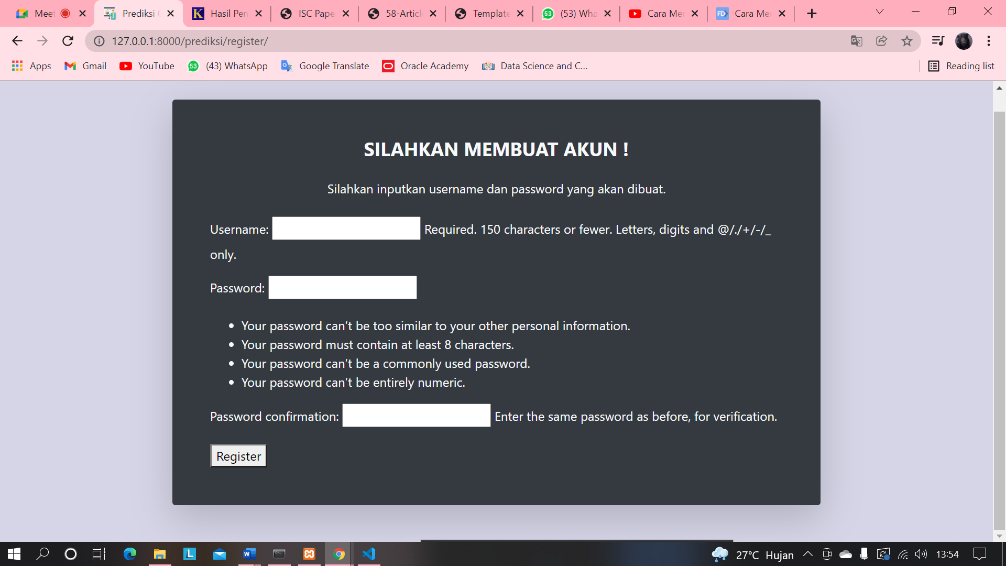
### **Implementasi Halaman *Login***



Gambar 22 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 23 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.

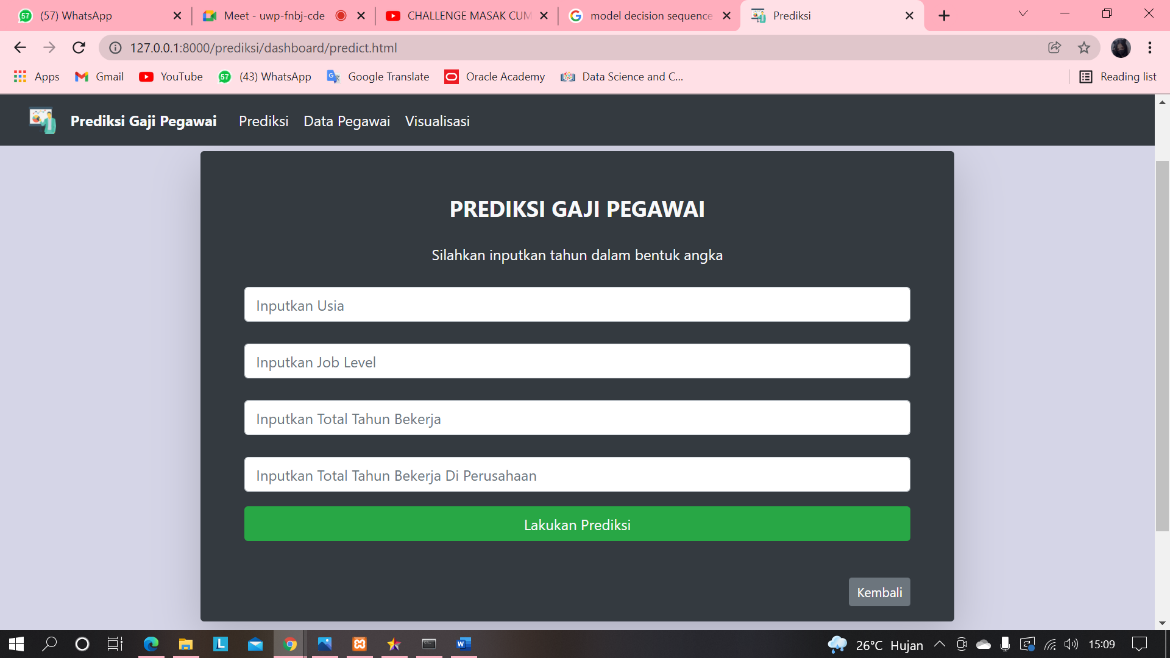
### **Implementasi Halaman *Dashboard***



Gambar 24 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.

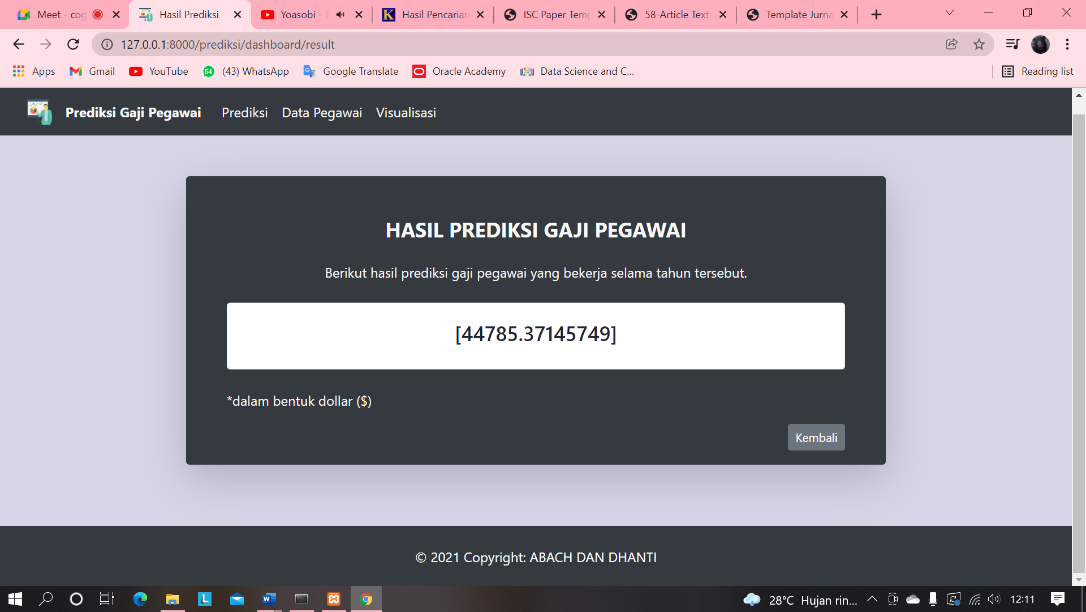
### **Implementasi Halaman Prediksi**



Gambar 25 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 berupa usia, job level, total tahun bekerja, dan total tahun bekerja di perusahan ke dalam form untuk mengetahui hasil prediksi gaji pegawai. Kemudian *user* dapat menekan button “Lakukan Prediksi” untuk melihat hasil prediksi berdasarkan tahun yang diinputkan.

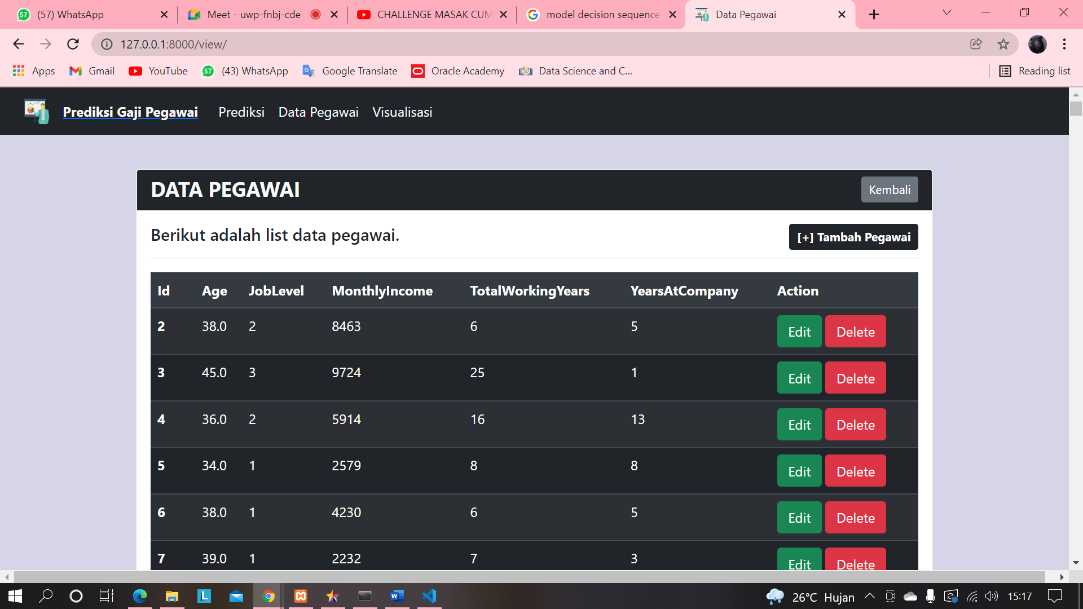
### **Implementasi Halaman Hasil Prediksi**



Gambar 26 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.

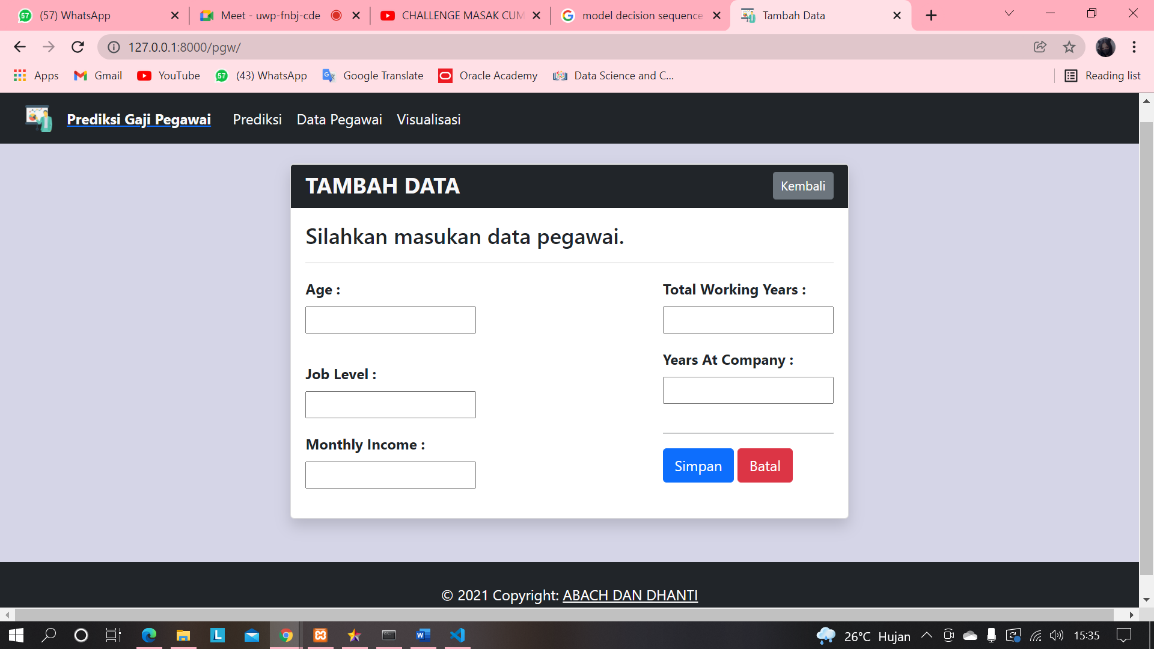
### **Implementasi Halaman Data Pegawai**



Gambar 27 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, Age, JobLevel, MonthlyIncome, dan Action. *Record* yang ditampilkan terdiri dari 439 *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.

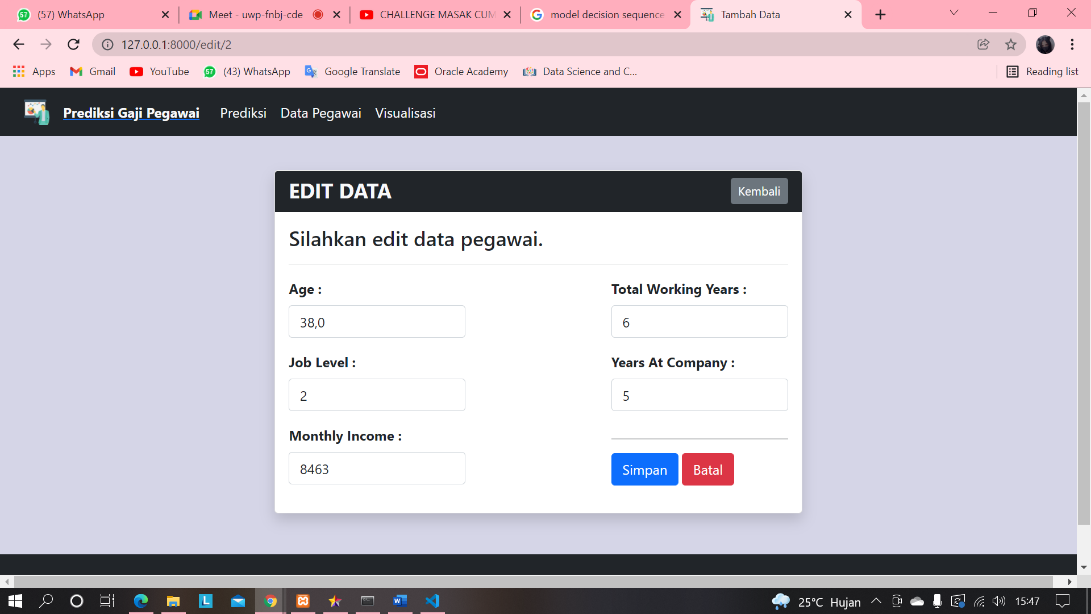
### **Implementasi Halaman Tambah Data Pegawai**



Gambar 28 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 Age, Job Level, Monthly Income, Total Working Years, dan Years At Company 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.

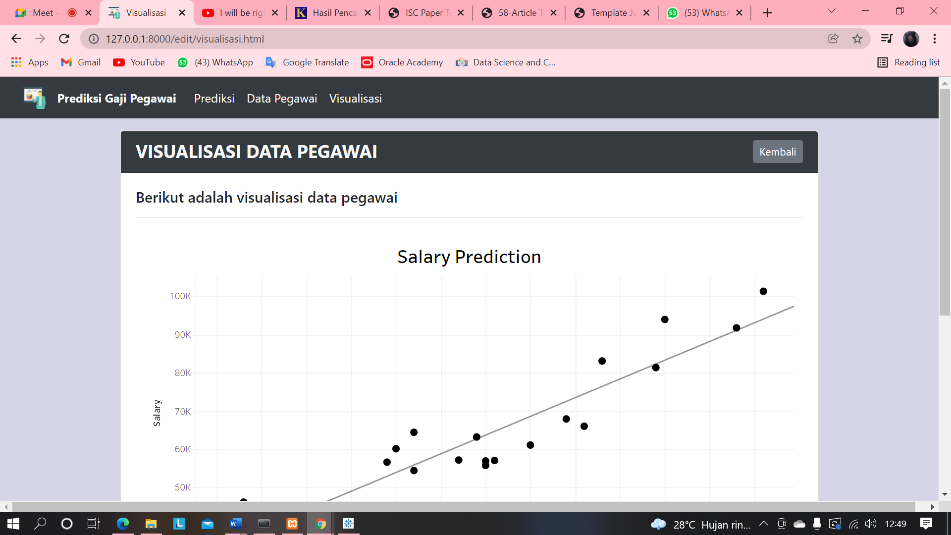
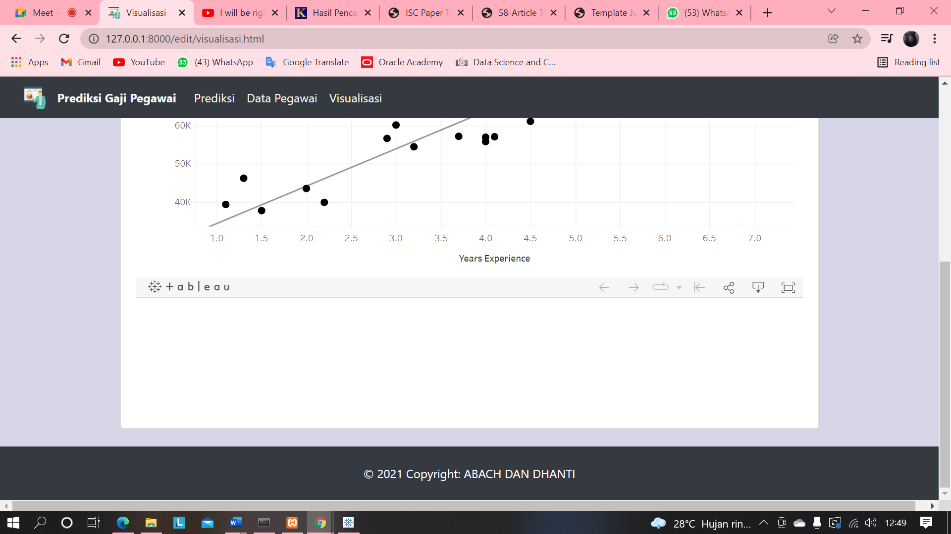
### **Implementasi Halaman Edit Data Pegawai**



Gambar 29 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 Age, Job Level, Monthly Income, Total Working Years, dan Years At Company pada form. Lalu *user* dapat menekan *button* Simpan untuk menyimpan data yang telah diubah. *User* dapat menekan *button* Batal untuk me-*reset* data yang diinputkan pada form.

### **Implementasi Halaman Visualisasi**



Gambar 30 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="Age" id="Age" class="form-control"  placeholder="Inputkan Usia" autocomplete="off" required ><br>

  <input type="integer" name="JobLevel" id="JobLevel" class="form-control"  placeholder="Inputkan Job Level" autocomplete="off" required ><br>

  <input type="integer" name="YearsAtCompany" id="YearsAtCompany" class="form-control"  placeholder="Inputkan Total Tahun Bekerja" autocomplete="off" required ><br>

  <input type="integer" name="TotalWorkingYears" id="TotalWorkingYears" class="form-control"  placeholder="Inputkan Total Tahun Bekerja Di Perusahaan" 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">Age</th>

      <th scope="col">JobLevel</th>

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

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

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

      <th scope="col">Action</th>

    </tr>

  </thead>

  {% for pegawai in pegawai %}

    <tr>

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

      <td>{{ pegawai.Age }}</td>

      <td>{{ pegawai.JobLevel }}</td>

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

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

      <td>{{ pegawai.YearsAtCompany }}</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="float-left">

     <div class="form-row">

      <div class="form-group col-md-6">

     <label for=" "><strong>Age :</strong></label>

       <div>{{ form.Age }}</div>

       </div>

      </div>

     <div class="form-row">

      <div class="form-group col-md-6">

       <label for=" "><strong>Job Level : </strong></label>

       <div>{{ form.JobLevel }}</div>

      </div>

     </div>

     <div class="form-row">

      <div class="form-group col-md-13">

       <label for=" "><strong>Monthly Income : </strong></label>

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

        </div>

      </div>

      </div>

       <div class="float-right">

      <div class="form-row">

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

     <label for=" "><strong>Total Working Years : </strong></label>

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

      </div>

      </div>

       <div class="form-row">

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

      <label for=" "><strong>Years At Company : </strong></label>

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

       </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="float-left">

    <div class="form-row">

    <div class="form-group col-md-15">

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

    <div>

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

   </div>

    </div>

   </div>

   <div class="form-row">

    <div class="form-group col-md-15">

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

    <div>

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

   </div>

    </div>

     </div>

     <div class="form-row">

     <div class="form-group col-md-15">

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

   <div>

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

   </div>

   </div>

     </div>

     </div>

     <div class="float-right">

     <div class="form-row">

     <div class="form-group col-md-15">

    <label for="jenis\_produk"><strong>Total Working Years : </strong></label>

     <div>

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

     </div>

     </div>

     </div>

     <div class="form-row">

    <div class="form-group col-md-15">

   <label for="jenis\_produk"><strong>Years At Company : </strong></label>

     <div>

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

      </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>

    </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] | import pandas as pd |
| [2] | from sqlalchemy import create\_engine |
| [3] | import pymysql |
| [4] | #mysql+pymysql://<username>:<password>@<host>/<dbname>[?<options>]  sqlEngine=create\_engine('mysql+pymysql://root:@127.0.0.1',pool\_recycle=3600) |
| [5] | dbConnection = sqlEngine.connect() |
| [6] | df = pd.read\_sql('SELECT \* FROM djpegawai.prediksi\_pegawai ', dbConnection) |
| [7] | pd.set\_option('display.expand\_frame\_repr', False) |
| [8] | print(df) |
| [9] | dbConnection.close() |
| [10] | df\_clean = df.drop('id', axis=1) |
| [11] | x = df2.iloc[:, :-1] |
| [12] | y = df2.iloc[:, -1] |

Keterangan :

* Line 1 : Untuk mengimport library pandas kemudian diinisialisasi dengan varibel pd
* Line 2 : Dari sqlalchemy diimport metode create\_engine
* Line 3 : Untuk mengimport library pymysql.
* Line 4 : Membuat engine database MySQL kemudian disimpan dalam variabel sqlEngine.
* Line 5 : Membuat koneksi dengan metode sqlEngine agar terhubung dengan database, kemudian simpan dalam variabel dbConnection.
* Line 6 : Membaca data pada database djpegawai tabel prediksi\_pegawai dan dikoneksikan dengan dbConnections, kemudian disimpan pada variabel df.
* Line 7 : Representasi dataframe keseluruhan
* Line 8 : Menampilkan data pada variabel df.
* Line 9 :
* Line 10 : Menghapus variabel id kemudian disimpan pada variabel df\_clean.
* Line 11 : Untuk menampilkan data yang terdapat dalam variabel df2.
* Line 12 : Membuat variabel X yang di dalamnya column yang diambil itu column Salary.
* Line 13 : Membuat variabel Y yang di dalamnya column yang diambil itu column YearsExperience

### **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.

|  |  |
| --- | --- |
| [13] | from sklearn.model\_selection import train\_test\_split |
| [14] | x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=1/3, random\_state=0) |
| [15] | #Modelling Machine Learning  from sklearn.linear\_model import LinearRegression |
| [16] | regressor = LinearRegression() |
| [17] | persamaan = regressor.fit(x\_train, y\_train) #training |
| [18] | print(regressor.coef\_) |
| [19] | print(regressor.intercept\_) |
| [20] | y\_pred = regressor.predict(x\_test) |

Keterangan :

* Line 13 : Dari scikit learn yang dimana menggunakan model selection untuk mengimport metode train\_test\_split.
* Line 14 : Membuat variabel x\_train, x\_test, y\_train, dan y\_test menggunakan metode train\_test\_split yang dimana variabel yang digunakan adalah x, y, dan untuk data testnya sebanyak 1/3 total data yang ada dengan pemilihan data secara acak.
* Line 15 : Dari scikit learn linear model diimport metode LinearRegression.
* Line 16 : Membuat variabel regressor yang isinya metode LinearRegression.
* Line 17 : Membuat variabel persamaan yang isinya terdapat method regressor.fit dengan parameternya x\_train dan y\_train.
* Line 18 : Mencari koefisien dari x menggunakan regressor, kemudian ditampilkan.
* Pada bagian Years Experience memiliki nilai koefisien 9087. Nilia tersebut memiliki arti bahwa tiap karyawan yang mengalami pertambahan satu tahun kerja, maka kenaikan gaji yang diterima sebesesar 9087.
* Line 19 : 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 menerima gaji sebesar 26611 per tahunnya.

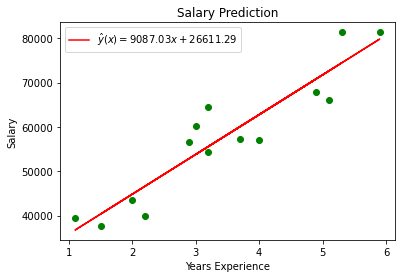
Berdasarkan nilai koefisien Years Experience dan Intersept didapat persamaan regresi linear berikut :

**Salary = 9087\*Years Experience + 26611**

* Line 20 : Menentukan variabel y\_pred yang didalamnya mencari prediksi mengggunakan variabel equation dengan parameternya x\_test.

1. Visualisasi Data Train

|  |  |
| --- | --- |
| [21] | import matplotlib.pyplot as plt |
| [22] | plt.scatter(x\_train, y\_train, color="green") |
| [23] | plt.plot(x\_train, regressor.predict(x\_train), color="red") |
| [24] | plt.title("Salary Prediction") |
| [25] | plt.xlabel("Years Experience") |
| [26] | plt.ylabel("Salary") |
| [27] | plt.legend(('$\hat{y}(x)=9087.03x+26611.29$',),loc='upper left') |
| [28] | plt.show() |



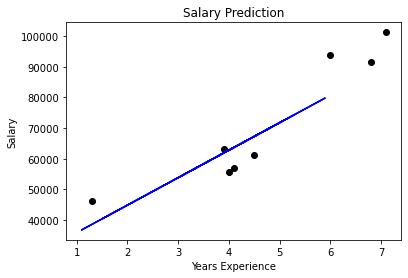
Gambar 31 Hasil Regresi Data Training

Keterangan :

* Line 21 : Mengimport library matplotlib.pyplot.
* Line 22 : Menampilkan titik/scatter x\_train dan y\_train berwarna hijau.
* Line 23 : Menampilkan plot/garis pada grafik berwarna merah.
* Line 24 : Membuat judul grafik dengan nama Salary vs Years Experience.
* Line 25 : Membuat label sumbu x dengan nama Years Experience.
* Line 26 : Membuat label sumbu y dengan nama Salary.
* Line 27 : Menampilkan fungsi linear y = 9087.03x+26611.29 pada grafik.
* Line 28 : Menampilkan visualisasi grafik.
* Line 29 : Dari scikit learn metrics diimport metode r2\_score.
* Line 30 : Menghitung r2\_score dengan parameternya x\_train dan y\_train, kemudian hasilnya disimpan pada variable r2.
* Line 31 : Menampilkan nilai pada variabel r2.

1. Visualisasi Data Test

|  |  |
| --- | --- |
| [29] | plt.scatter(x\_test, y\_test, color="black") |
| [30] | plt.plot(x\_train, regressor.predict(x\_train), color="blue") |
| [31] | plt.title("Salary Prediction") |
| [32] | plt.xlabel("Years Experience") |
| [33] | plt.ylabel("Salary") |
| [34] | plt.show() |



Gambar 32 Hasil Regresi Data Test

Keterangan :

* Line 29 : Menampilkan titik/scatter x\_test dan y\_test berwarna hitam.
* Line 30 : Menampilkan plot/garis pada grafik berwarna biru.
* Line 31 : Membuat judul grafik dengan nama Salary Prediction.
* Line 32 : Membuat label sumbu x dengan nama Years Experience.
* Line 33 : Membuat label sumbu y dengan nama Salary.
* Line 34 : Menampilkan visualisasi grafik.

### **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.

|  |  |
| --- | --- |
| [35] | from sklearn.metrics import r2\_score |
| [36] | r2 = r2\_score (x\_train, y\_train) |
| [37] | print ("Nilai R2 adalah ", r2) |

Keterangan :

* Line 35 : Dari scikit learn metrics diimport metode r2\_score.
* Line 36 : Menghitung r2\_score dengan parameternya x\_train dan y\_train, kemudian hasilnya disimpan pada variable r2.
* Line 37 : Menampilkan nilai pada variabel r2.

1. Tabel ANOVA

|  |  |
| --- | --- |
| [38] | import numpy as np |
| [39] | import pandas as pd |
| [40] | import scipy |
| [41] | from scipy import stats |
| [42] | def ANOVATAB(y,yhat,n,m):  dfn=n  dfd=m-n-1  ybar=np.average(y)    SSR = sum((yhat-ybar)\*\*2)  SSE = sum((y-yhat)\*\*2)  print (SSE)  print(dfd)  SST = sum((y-ybar)\*\*2)  MSR = SSR/dfn  MSE = SSE/dfd    Fs = MSR/MSE  ks =1-scipy.stats.f.cdf(Fs,dfn,dfd)  data\_table = {  'SS' : [SSR,SSE,SST],  'df' : [dfn,dfd,m-1],  'MS' : [MSR,MSE,'-'],  'Fs' : [Fs,'-','-'],  'pval' : [ks, '-', '-']  }  return pd.DataFrame(data\_table) |
| [43] | n=1  m=len(y\_pred)  print('TABEL ANOVA')  result=ANOVATAB(y\_test,y\_pred,n,m)  result |

Keterangan :

* Line 38 : Untuk mengimport libraty numpy kemudian diinisialisasi dengan variabel np.
* Line 39 : Untuk mengimpoty library pandas kemudian diinisialisasi dengan variabel pd.
* Line 40 : Untuk mengimport library scipy.
* Line 41 : Dari Scipy diimport metode stats.
* Line 42 : Untuk membuat rumus perhitungan pada tabel ANOVA.

Tabel ANOVA (Analysis of Variance) adalah tabel untuk menguji penerimaan (acceptability) model dari perspektif statistic dalam bentuk analisis sumber keragaman. ANOVA ini sering juga diterjemahkan sebagai analisis ragam dan output dari ANOVA adalam penjelasan uji regresi. Pada tabel tersebut tertera nilai dari SS, df, MS, FS, dan Pval.

1. SS (Sum of Square)

Kolom SS diperoleh dari penjumlahan kuadrat dari prediksi variabel terikat (permintaan) dikurangi dengan nilai rata-rata permintaan dari dara sebenarnya. Jadi secara manual kit acari dulu rata-rata permintaan dari data asli kita. Kemudian masing-masing prediksi permintaan (lihat pada tabel residual output) dikurangi dengan rata-rata tersebut kemudian dikuadratkan. Selanjutnya, seluruh hasil perhitungan tersebut dijumlahkan.

1. Df (Degree of Freedom)

Df atau derata bebas dari total adalah n-1, dimana n adalah banyaknya observasi.

1. MS (Mean of Square)

MS atau rata-rata jumlah kuadrat adalah hasil bagi antara kolom SS dengan kolom df. Dari perhitungan MS ini, selanjutnya bagi MS Regresi dengan MS Residual yang kemudian akan mendapatkan nilai F.

1. 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 0.05, hal ini menandakan ia signifikan melakukan prediksi variabel dependen (Salary).

1. 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.

* Line 43 : Untuk menampilkan hasil perhitungan dari tabel ANOVA.

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

1. Prediksi Gaji

|  |  |
| --- | --- |
| [44] | salary\_pred = regressor.predict([[7]]) |
| [45] | print("Gaji yang terprediksi pada pegawai yang bekerja sepanjang tahun tersebut adalah ",salary\_pred) |

Keterangan :

* Line 44 : Menentukan prediksi gaji dengan parameter YearsExperience nya 7 tahun menggunakan metode pada variabel regressor, kemudian simpan di variabel salary\_pred.
* Line 45 : Menampilkan nilai dari variabel salary\_pred.