

Final Project Data Mining
“Laporan Final Project: Klasifikasi Kelayakan Job Listing
(Studi kasus: Jobstreets)”



Dosen Pengampu :

Amalia Anjani Arifiyanti, S.Kom., M.Kom.

Disusun oleh :

Kevin Yohanes Wuryanto	21082010193
Dewangga Nanda Arjuna	21082010201
Bima Mukti Wibowo	210820102244

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PEMBANGUNAN NASIONAL “VETERAN”
JAWA TIMUR
2024

DAFTAR ISI

DAFTAR ISI.....	2
FINAL PROJECT DATA MINING.....	3
BAB I PENDAHULUAN.....	3
1.1. Latar Belakang.....	3
1.2. Pengumpulan Data.....	3
BAB II.....	4
HASIL DAN PEMBAHASAN.....	4
2.1. Scraping.....	4
2.2. Preprocessing.....	6
2.3. Exploratory.....	6
2.4. Pemodelan.....	10
BAB III.....	25
KESIMPULAN.....	25

FINAL PROJECT DATA MINING

BAB I PENDAHULUAN

1.1. Latar Belakang

Dalam era digital saat ini, pencarian kerja telah mengalami transformasi signifikan dengan hadirnya platform online seperti Jobstreet. Platform ini memfasilitasi pertemuan antara perusahaan dan pencari kerja dalam ruang digital, sehingga proses rekrutmen dan pengajuan lamaran kerja menjadi lebih praktis dan efisien. Namun, dibalik kemudahan tersebut, tantangan baru muncul, terutama terkait transparansi informasi mengenai kelayakan gaji untuk posisi pekerjaan tertentu di lokasi yang berbeda.

Bagi pencari kerja, memahami kisaran gaji yang layak untuk suatu pekerjaan berdasarkan lokasi menjadi krusial dalam membuat keputusan karier yang tepat. Kondisi ini menyulitkan analisis langsung dan membatasi wawasan pencari kerja tentang nilai pasar suatu pekerjaan di berbagai wilayah.

Oleh karena itu, penelitian ini bertujuan untuk menganalisis kelayakan pekerjaan berdasarkan gaji, pekerjaan dan lokasi yang ditawarkan. Proses ini meliputi ekstraksi informasi, pembersihan data, dan analisis statistik untuk mengolah data gaji yang tersedia secara sistematis. Dengan pendekatan ini, penelitian diharapkan dapat memberikan wawasan yang lebih transparan dan akurat tentang kelayakan gaji, baik bagi pencari kerja sebagai referensi negosiasi gaji, maupun bagi perusahaan untuk menyusun penawaran gaji yang kompetitif sesuai dengan lokasi dan posisi pekerjaan.

1.2. Pengumpulan Data

Data yang digunakan dalam penelitian ini diperoleh dari platform Jobstreet, yang merupakan salah satu portal pencarian kerja terkemuka di Asia. Data ini mencakup informasi terkait kisaran gaji, posisi pekerjaan, lokasi, dan industri. Adapun proses pengumpulan data dilakukan dengan teknik **web scraping** menggunakan API dari website jobstreet.

BAB II

HASIL DAN PEMBAHASAN

2.1. Scraping

1. install Library Selenium

Selenium adalah library open-source yang dapat digunakan untuk web scraping, yaitu pengumpulan data dari internet secara otomatis dalam skala besar.

Script :

```
[20] !pip install selenium
```

2. Import Libraries

Script :

```
[21] from selenium import webdriver
      from selenium.webdriver.common.by import By
      from selenium.common.exceptions import NoSuchElementException
      from selenium.common.exceptions import StaleElementReferenceException
      from numpy import nan
      import pandas as pd
      import time
```

3. Getting Updates and Chromium.

Script :

```
!apt-get update
!apt-get install chromium-driver
```

4. Configuration for Web Driver.

Script :

```
[23] def web_driver():
      options = webdriver.ChromeOptions()
      options.add_argument("--verbose")
      options.add_argument("--no-sandbox")
      options.add_argument("--headless")
      options.add_argument("--disable-gpu")
      options.add_argument("--window-size=1920,1200")
      options.add_argument("--disable-dev-shm-usage")
      driver = webdriver.Chrome(options=options)
      return driver
```

5. Declare Driver.

Script :

```
[32] driver = web_driver()
```

6. Getting HTML for Driver with URL

Script :

```
[33] driver.get("https://id.jobstreet.com/id/jobs")
```

7. Pathing Specific Class in HTML

Script :

```
containers = driver.find_elements(By.XPATH,
'//article[@class="_1ns0a640 _1ns0a641 c4q8kx8n c4q8kx8o
c4q8kx7j c4q8kx7k c4q8kxav c4q8kxaw c4q8kx9r c4q8kx9s c4q8kxh
c4q8kx67 c4q8kx5f _1t6dkmpb _1t6dkmp9 _1t6dkmpa _1ash1w718
_1ash1w71b c4q8kx33 c4q8kx36"]')
```

8. Scraping + Exporting.

Script :

9. Displays data that has been scraped and exported.

Script :

```
[36] df = pd.read_csv('/content/job_data.csv')
df
```

Output :

	Title	Company	Location	Salary
0	HEAD WORKSHOP INTERIOR FURNITURE	Palais Construction	Tangerang, Banten	Rp 9.000.000 – Rp 11.000.000 per month
1	Sales Executive	PT.MULIA DUTA INDONESIA	Jakarta Timur, Jakarta Raya	Rp 3.000.000 – Rp 4.000.000 per month
2	Purchasing Staff Restoran	PT Kideta Boga Multirasa	Jakarta Utara, Jakarta Raya	Rp 5.000.000 – Rp 6.000.000 per month
3	Customer Relation Officer Jakarta	EF English First Swara Group	Jakarta Barat, Jakarta Raya	Rp 5.000.000 – Rp 7.000.000 per month
4	Operator Produksi (Percetakan)	Tumbleryuk	Jakarta Barat, Jakarta Raya	Rp 2.500.000 – Rp 3.500.000 per month
...
2292	SENIOR Product Designer (UI/UX)	Design Elite	Malang, Jawa Timur	Rp 8.000.000 – Rp 10.000.000 per month
2293	Creative Design Manager	PT winetobe Bali Indonesia	Bali	Rp 12.000.000 – Rp 18.000.000 per month
2294	Staff Schedule Proyek Konstruksi (Penempatan J...	PT BRINGIN KARYA SEJAHTERA	Jakarta Selatan, Jakarta Raya	Rp 5.500.000 – Rp 6.000.000 per month
2295	Programmer	Yayasan Rumah Sakit Charitas	Sumatera Selatan	Rp 4.000.000 – Rp 6.000.000 per month
2296	IT Staff	PT Crown Worldwide Indonesia	Tangerang Selatan, Banten	Rp 5.000.000 – Rp 6.000.000 per month
2297 rows x 4 columns				

2.2. Preprocessing

Pada tahap preprocessing dilakukan pelabelan manual untuk data yang telah didapatkan. Pelabelan dilakukan dengan memberikan Label penilaian “Layak” dan “Tidak Layak” berdasarkan penilaian subjektif. berikut adalah contoh pelabelan yang dilakukan :

No	Title	Company	Location	Salary	Kelayakan
1	HEAD WORKSHOP INTERIOR FURNITURE	Palais Construction	Tangerang, Banten	Rp 9.000.000 - Rp 11.000.000 per month	Layak
2	Purchasing Staff Restoran	PT Kideta Boga Multirasa	Jakarta Utara, Jakarta Raya	Rp 5.000.000 - Rp 6.000.000 per month	Tidak Layak
3	Customer Relation Officer Jakarta	EF English First Swara Group	Jakarta Barat, Jakarta Raya	Rp 5.000.000 - Rp 7.000.000 per month	Tidak Layak
4.	dst...	dst...	dst...	dst...	dst...

2.3. Exploratory

a. Import Libraries

Script :

```
[3] import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

b. Import Dataset (job_data.csv)

Script :

```
[4] df = pd.read_csv('/content/job_data.csv')
df
```

Output :

	Title	Company	Location	Salary
0	HEAD WORKSHOP INTERIOR FURNITURE	Palais Construction	Tangerang, Banten	Rp 9.000.000 – Rp 11.000.000 per month
1	Purchasing Staff Restoran	PT Kideta Boga Multirasa	Jakarta Utara, Jakarta Raya	Rp 5.000.000 – Rp 6.000.000 per month
2	Customer Relation Officer Jakarta	EF English First Swara Group	Jakarta Barat, Jakarta Raya	Rp 5.000.000 – Rp 7.000.000 per month
3	Operator Produksi (Percetakan)	Tumbleryuk	Jakarta Barat, Jakarta Raya	Rp 2.500.000 – Rp 3.500.000 per month
4	Telesales/Telemarketing (Cabang Manado)	PT. JAYA BERSAMA SAPUTRA PERKASA	Manado, Sulawesi Utara	Rp 3.000.000 – Rp 4.500.000 per month
...
2266	Financial Advisor - Area Tangerang Kota	PT AXA Mandiri Financial Services	Tangerang Selatan, Banten	Rp 3,200,000 – Rp 4,800,000 per month
2267	Interior Drafter	PT Jensen Group Bali	Kuta Utara, Bali	Rp 4,000,000 – Rp 6,000,000 per month
2268	FINANCE ACCOUNTING & TAX	PT Indobeef Indonesia	Jakarta Selatan, Jakarta Raya	Rp 5.000.000 – Rp 6.000.000 per month
2269	Accounting	PT Mansoer Dasa Bersaudara	Bandung, Jawa Barat	Rp 4.000.000 – Rp 6.000.000 per month
2270	Finance & Accounting Supervisor	QQ KOPITIAM	Jakarta Selatan, Jakarta Raya	Rp 6.000.000 – Rp 8.000.000 per month

c. EDA Describe

Script :

```
df.describe()
```

Output :

	Title	Company	Location	Salary
count	2271	2271	2271	2271
unique	1982	1540	253	633
top	Sales Executive	Pengiklan Anonim	Jakarta Raya	Rp 5.000.000 – Rp 7.000.000 per month
freq	22	113	256	89

d. EDA Check Nulls

Script :

```
df.isnull().sum()
```

Output :

	0
Title	0
Company	0
Location	0
Salary	0

e. Attribute Variance

Script :

```
[7] print(df['Title'].value_counts(), "\n")
print(df['Company'].value_counts(), "\n")
print(df['Location'].value_counts(), "\n")
print(df['Salary'].value_counts(), "\n")
```

Output

Title		Location	
Sales Executive	22	Jakarta Raya	256
Host Live Streaming	15	Jakarta Utara, Jakarta Raya	229
Mandarin Translator	10	Jakarta Selatan, Jakarta Raya	205
Content Creator	9	Jakarta Barat, Jakarta Raya	197
Sales Manager	9	Jakarta Pusat, Jakarta Raya	120
--	--	---	---
Team Leader Field Collection Support (Bogor & Bekasi)	1	Telukjambe Barat, Jawa Barat	1
International Health Insurance Administration	1	Cilincing, Jakarta Raya	1
ASISTEN OWNER (MANDARIN SPEAKER)	1	Cileungsi, Jawa Barat	1
Sous Chef (Kitchen) The Coach Restaurant	1	Tanah Bumbu, Kalimantan Selatan	1
Finance & Accounting Supervisor	1	Kuta Utara, Bali	1
Name: count, Length: 1982, dtype: int64		Name: count, Length: 253, dtype: int64	
Company		Salary	
Pengiklan Anonim	113	Rp 5.000.000 - Rp 7.000.000 per month	89
PT. PERSOLKELLY Recruitment Indonesia	23	Rp 10.000.000 - Rp 15.000.000 per month	87
RGF HR Agent Indonesia	22	Rp 4.000.000 - Rp 6.000.000 per month	68
PT Cilia Prima Utama Makmur	18	Rp 5.000.000 - Rp 6.000.000 per month	67
PT Home Credit Indonesia	9	Rp 7.000.000 - Rp 10.000.000 per month	65
---	---	--	--
Facetology	1	Rp 13.300.000 - Rp 19.500.000 per month	1
PERSEORAN TERBATAS - BADAN CITRA KENCANA SOLUSINDO	1	Rp 5.100.000 - Rp 5.500.000 per month	1
PT Aditya Mandiri Sejahtera	1	Rp 3.800.000 - Rp 4.100.000 per month	1
Chesindo Group	1	Rp 7,500,000 - Rp 9,000,000 per month	1
QQ KOPITIAM	1	Rp 3,200,000 - Rp 4,800,000 per month	1
Name: count, Length: 1540, dtype: int64		Name: count, Length: 633, dtype: int64	

f. View Duplicate

Script :

```
[8] df = pd.read_csv('/content/job_data.csv')
duplicate_rows = df[df.duplicated()]
num_duplicates = len(duplicate_rows)
print(f"Number of duplicate rows in 'job_data.csv': {num_duplicates}")

print("\nDuplicate Rows:")
duplicate_rows
```

Output :

Number of duplicate rows in 'job_data.csv': 1

Duplicate Rows:

	Title	Company	Location	Salary
1866	Financial Marketing	PT Mitra Cepat Sukses	Jakarta Pusat, Jakarta Raya	Rp 6.000.000 – Rp 9.000.000 per year

	Title	Company	Location	Salary
1866	Financial Marketing	PT Mitra Cepat Sukses	Jakarta Pusat, Jakarta Raya	Rp 6.000.000 – Rp 9.000.000 per year

	Title	Company	Location	Salary
1866	Financial Marketing	PT Mitra Cepat Sukses	Jakarta Pusat, Jakarta Raya	Rp 6.000.000 – Rp 9.000.000 per year

g. Word Cloud

Script :

```
[9] import pandas as pd
from wordcloud import WordCloud
import matplotlib.pyplot as plt

df = pd.read_csv('/content/job_data.csv')

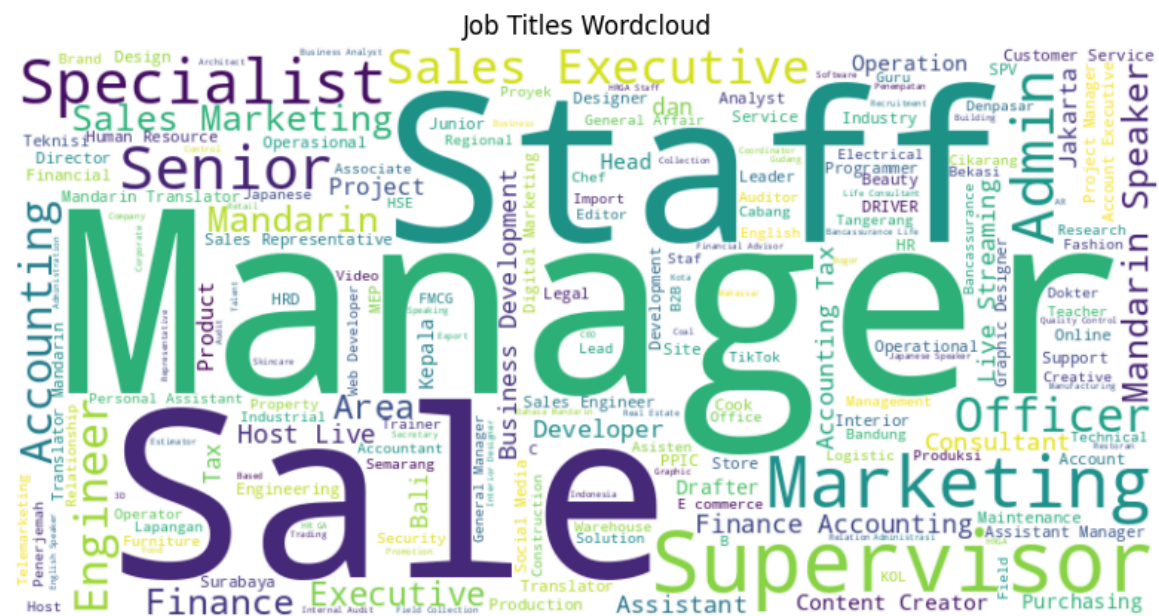
def create_wordcloud(text, title):
    wordcloud = WordCloud(width=800, height=400, background_color='white').generate(text)
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis("off")
    plt.title(title)
    plt.show()

text = ' '.join(df['Title'].astype(str).tolist())
create_wordcloud(text, "Job Titles Wordcloud")

text = ' '.join(df['Company'].astype(str).tolist())
create_wordcloud(text, "Company Names Wordcloud")

text = ' '.join(df['Location'].astype(str).tolist())
create_wordcloud(text, "Locations Wordcloud")
```

Output :



2.4. Pemodelan

a. Libraries

Script :

```
[1] # Base
import pandas as pd

# Split
from sklearn.model_selection import train_test_split

# Vector
from sklearn.feature_extraction.text import TfidfVectorizer, TfidfTransformer

# Model
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier

# Evaluation
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

b. Pre-modelling (Skenario 1 - 70:30)

Script :

```
[2] data_train = pd.read_csv('/content/train_balance.csv')

x_train = data_train.drop('Kelayakan', axis=1)
y_train = data_train['Kelayakan']

X_train, X_test, y_train, y_test = train_test_split(
    x_train, y_train, test_size=0.3, random_state=33)

X_train_combined = X_train.astype(str).apply(" ".join, axis=1)
X_test_combined = X_test.astype(str).apply(" ".join, axis=1)

vectorizer = TfidfVectorizer(use_idf=True, strip_accents='ascii')
X_train_tfidf = vectorizer.fit_transform(X_train_combined)
X_test_tfidf = vectorizer.transform(X_test_combined)
```

c. Pre-modelling (Skenario 2 - 80:20)

Script :

```
[206] data_train2 = pd.read_csv('/content/dataset_balance.csv')

X_train2 = data_train2.drop('Kelayakan', axis=1)
y_train2 = data_train2['Kelayakan']

X_train2, X_test2, y_train2, y_test2 = train_test_split(
    X_train2, y_train2, test_size=0.2, random_state=33)

X_train_combined2 = X_train2.astype(str).apply(" ".join, axis=1)
X_test_combined2 = X_test2.astype(str).apply(" ".join, axis=1)

vectorizer2 = TfidfVectorizer(use_idf=True, strip_accents='ascii')
X_train_tfidf2 = vectorizer2.fit_transform(X_train_combined2)
X_test_tfidf2 = vectorizer2.transform(X_test_combined2)
```

d. Skenario 1 70:30

i. Naive bayes - 70:30

Script :

```
model_nb = MultinomialNB()
model_nb.fit(X_train_tfidf, y_train)

y_pred_NB = model_nb.predict(X_test_tfidf)

accuracy = accuracy_score(y_test, y_pred_NB)
conf_matrix = confusion_matrix(y_test, y_pred_NB)
class_report = classification_report(y_test, y_pred_NB)
print("Accuracy:", accuracy)
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(class_report)
```

output :

```

Accuracy: 0.52
Confusion Matrix:
[[56 25]
 [47 22]]
Classification Report:

```

	precision	recall	f1-score	support
Layak	0.54	0.69	0.61	81
Tidak Layak	0.47	0.32	0.38	69
accuracy			0.52	150
macro avg	0.51	0.51	0.49	150
weighted avg	0.51	0.52	0.50	150

ii. Support Vector Machine (SVM) - 70:30

Script :

```

[4] # Train an SVM model
model_svm = SVC(kernel='linear')
model_svm.fit(X_train_tfidf, y_train)

# Make predictions on the validation set
y_pred_SVM = model_svm.predict(X_test_tfidf)

# # Evaluate the model
print("Validation Set Performance:")
print("Accuracy:", accuracy_score(y_test, y_pred_SVM))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_SVM))
print("Classification Report:\n", classification_report(y_test, y_pred_SVM))

```

output :

```

Validation Set Performance:
Accuracy: 0.66
Confusion Matrix:
[[65 16]
 [35 34]]
Classification Report:

```

	precision	recall	f1-score	support
Layak	0.65	0.80	0.72	81
Tidak Layak	0.68	0.49	0.57	69
accuracy			0.66	150
macro avg	0.67	0.65	0.64	150
weighted avg	0.66	0.66	0.65	150

iii. Decision Tree - 70:30

Script :

```
[ ] model_nb = MultinomialNB()
    model_nb.fit(X_train_tfidf, y_train)

    y_pred_NB = model_nb.predict(X_test_tfidf)

    accuracy = accuracy_score(y_test, y_pred_NB)
    conf_matrix = confusion_matrix(y_test, y_pred_NB)
    class_report = classification_report(y_test, y_pred_NB)
    print("Accuracy:", accuracy)
    print("Confusion Matrix:")
    print(conf_matrix)
    print("Classification Report:")
    print(class_report)
```

output :

```
Validation Set Performance:
Accuracy: 0.5733333333333334
Confusion Matrix:
[[50 31]
 [33 36]]
Classification Report:
              precision    recall  f1-score   support

    Layak         0.60      0.62      0.61         81
   Tidak Layak    0.54      0.52      0.53         69

   accuracy              0.57              150
  macro avg              0.57              150
weighted avg              0.57              150
```

e. Skenario 2 - 80:20

1. Pemodelan (+ Evaluasi)

a. Naive Bayes - 80:20

Script :

```
[13] model_nb2 = MultinomialNB()
      model_nb2.fit(X_train_tfidf2, y_train2)

      y_pred_NB2 = model_nb2.predict(X_test_tfidf2)

      print("Training Data Evaluation:")
      print(classification_report(y_test2, y_pred_NB2))
```

Output:

Training Data Evaluation:		precision	recall	f1-score	support
Layak		0.57	0.70	0.63	83
Tidak Layak		0.48	0.34	0.40	67
accuracy				0.54	150
macro avg		0.52	0.52	0.51	150
weighted avg		0.53	0.54	0.53	150

b. Support Vector Machine (SVM) - 80:20

Script :

```
[14] model_svm2 = SVC(kernel='linear')
      model_svm2.fit(X_train_tfidf2, y_train2)

      y_pred_SVM2 = model_svm2.predict(X_test_tfidf2)

      print("Training Data Evaluation (SVM):")
      print(classification_report(y_test2, y_pred_SVM2))
```

Output:

Training Data		Evaluation (SVM):			
		precision	recall	f1-score	support
	Layak	0.65	0.86	0.74	83
	Tidak Layak	0.70	0.42	0.52	67
	accuracy			0.66	150
	macro avg	0.67	0.64	0.63	150
	weighted avg	0.67	0.66	0.64	150

c. Decision Tree - 80:20

Script :

```
[ ] # Model Decision Tree
    model_dt2 = DecisionTreeClassifier(random_state=42)
    model_dt2.fit(X_train_tfidf2, y_train2)

    # Prediksi dengan Decision Tree
    y_pred_DT2 = model_dt2.predict(X_test_tfidf2)

    print("Training Data Evaluation (Decision Tree):")
    print(classification_report(y_test2, y_pred_DT2))
```

Output:

Training Data		Evaluation (Decision Tree):			
		precision	recall	f1-score	support
	Layak	0.60	0.60	0.60	83
	Tidak Layak	0.50	0.49	0.50	67
	accuracy			0.55	150
	macro avg	0.55	0.55	0.55	150
	weighted avg	0.55	0.55	0.55	150

2. ROC-AUC

Script :

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer

# Models
models = {
    "NB_Skenario1": MultinomialNB(),
    "SVM_Skenario1": SVC(kernel='linear', probability=True),
    "DT_Skenario1": DecisionTreeClassifier(random_state=42),
    "NB_Skenario2": MultinomialNB(),
    "SVM_Skenario2": SVC(kernel='linear', probability=True),
    "DT_Skenario2": DecisionTreeClassifier(random_state=42)
}

# Label encoding
label_encoder1 = LabelEncoder()
y_train1_encoded = label_encoder1.fit_transform(y_train)
y_test1_encoded = label_encoder1.transform(y_test)

label_encoder2 = LabelEncoder()
y_train2_encoded = label_encoder2.fit_transform(y_train2)
y_test2_encoded = label_encoder2.transform(y_test2)
```



```

# Models
models = {
    "NB_Skenario1": MultinomialNB(),
    "SVM_Skenario1": SVC(kernel='linear', probability=True),
    "DT_Skenario1": DecisionTreeClassifier(random_state=42),
    "NB_Skenario2": MultinomialNB(),
    "SVM_Skenario2": SVC(kernel='linear', probability=True),
    "DT_Skenario2": DecisionTreeClassifier(random_state=42)
}

# Fit models dan prediksi probabilitas untuk masing-masing skenario
predictions = {}
for model_name, model in models.items():
    if "Skenario1" in model_name:
        # Skenario 1 (70:30)
        model.fit(X_train_tfidf, y_train1_encoded)
        predictions[model_name] = model.predict_proba(X_test_tfidf)[:, 1]
    else:
        # Skenario 2 (80:20)
        model.fit(X_train_tfidf2, y_train2_encoded)
        predictions[model_name] = model.predict_proba(X_test_tfidf2)[:, 1]

# ROC-AUC plotting
plt.figure(figsize=(12, 8))
for model_name, y_pred in predictions.items():
    if "Skenario1" in model_name:
        y_test = y_test1_encoded
        label = "Skenario 1 (70:30)"
    else:
        y_test = y_test2_encoded
        label = "Skenario 2 (80:20)"

    fpr, tpr, _ = roc_curve(y_test, y_pred)
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr, tpr, lw=2, label=f'{model_name} ({label}) (AUC = {roc_auc:.4f})')

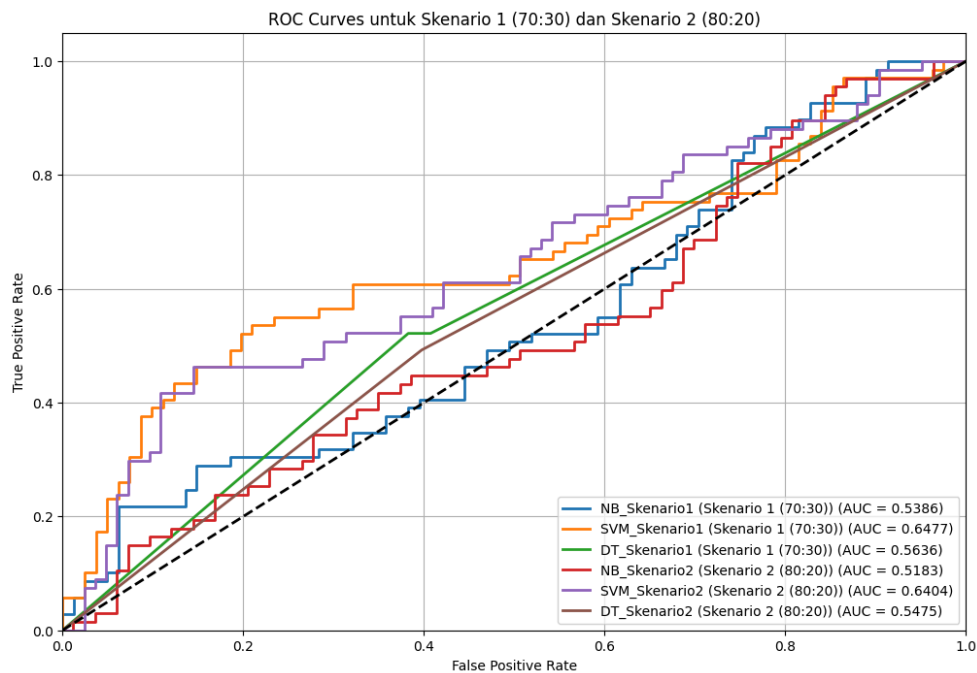
# Plot random guess line
plt.plot([0, 1], [0, 1], 'k--', lw=2)

# Add titles and labels
plt.title('ROC Curves untuk Skenario 1 (70:30) dan Skenario 2 (80:20)')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.legend(loc='lower right')
plt.grid()

# Show plot
plt.show()

```

Output:



f. Predict the actual Test File CSV

1. Skenario 1 - menggunakan SVM

Script :

```
[17] data_test = pd.read_csv('/content/data_fix.csv')

X_test_combined = data_test.astype(str).apply(" ".join, axis=1)

X_test_tfidf = vectorizer.transform(X_test_combined)

# PERINGATAN PILIH SALAH SATU!!!!!!!
# Naive Bayes (Multinomial)
# y_pred_test = model_nb.predict(X_test_tfidf)

# SVM
# y_pred_test = model_svm.predict(X_test_tfidf)

# Decision Tree
y_pred_test = model_dt.predict(X_test_tfidf)

data_test['Kelayakan'] = y_pred_test
data_test['Kelayakan'].value_counts()

# Exporting
# data_test.to_csv('/content/streamlit.csv', index=False)
```

Output:

	count
Kelayakan	
Layak	826
Tidak Layak	651

2. Skenario 2

Script :

```
[13] data_test2 = pd.read_csv('/content/data_fix.csv')

X_test_combined2 = data_test2.astype(str).apply(" ".join, axis=

X_test_tfidf2 = vectorizer2.transform(X_test_combined2)

# PERINGATAN PILIH SALAH SATU!!!!!!!!!!
# Naive Bayes (Multinomial)
# y_pred_test2 = model_nb2.predict(X_test_tfidf2)

# SVM
# y_pred_test2 = model_svm2.predict(X_test_tfidf2)

# Decision Tree
y_pred_test2 = model_dt2.predict(X_test_tfidf2)

data_test2['Kelayakan'] = y_pred_test2
data_test2['Kelayakan'].value_counts()
```

Output:

	count
Kelayakan	
Layak	827
Tidak Layak	650

2.5. Deployment

1. Import + Training + Accuracy

Script :

```
import pickle
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
import pandas as pd

# Load dataset
df = pd.read_csv('/content/streamlit.csv')

# Ubah kolom target menjadi biner
df['Kelayakan'] = df['Kelayakan'].map({'Tidak Layak': 0, 'Layak': 1})

# Gabungkan semua kolom fitur menjadi satu kolom teks
df['combined_text'] = df.iloc[:, :-1].astype(str).agg(' '.join, axis=1)

# Pisahkan fitur dan target
X = df['combined_text'] # Gabungan teks fitur
y = df['Kelayakan']     # Kolom target

# Vectorize data menggunakan TF-IDF
vectorizer = TfidfVectorizer()
X_tfidf = vectorizer.fit_transform(X) # Hasil berupa matriks sparse

# Simpan vectorizer
with open('tfidf_vectorizer.pkl', 'wb') as vec_file:
    pickle.dump(vectorizer, vec_file)

# Split data
X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y, test_size=0.3, random_state=0)

# Inisialisasi dan fit model SVC
classifier = DecisionTreeClassifier(random_state=42)
# classifier = SVC(kernel='linear')
# classifier = MultinomialNB()
classifier.fit(X_train, y_train)

# Prediksi
y_pred = classifier.predict(X_test)

# Evaluasi akurasi
score = accuracy_score(y_test, y_pred)
print(f'Akurasi: {score:.4f}')
```

Output:

```
Akurasi: 0.8423
```

2. Import Libraries

Script :

```
[27] import pickle

pickle_out = open("classifier.pkl", "wb")
pickle.dump(classifier, pickle_out)
pickle_out.close()
```

3. Install Streamlit

Script :

```
[28] !pip install streamlit
```

Output:

```
Collecting streamlit
  Downloading streamlit-1.41.1-py2.py3-none-any.whl.metadata (8.5 kB)
Requirement already satisfied: altair<6,>=4.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (5.5.0)
Requirement already satisfied: blinker<2,>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (1.9.0)
Requirement already satisfied: cachetools<6,>=4.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (5.5.0)
Requirement already satisfied: click<9,>=7.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (8.1.7)
Requirement already satisfied: numpy<3,>=1.23 in /usr/local/lib/python3.10/dist-packages (from streamlit) (1.26.4)
Requirement already satisfied: packaging<25,>=20 in /usr/local/lib/python3.10/dist-packages (from streamlit) (24.2)
Requirement already satisfied: pandas<3,>=1.4.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (2.2.2)
Requirement already satisfied: pillow<12,>=7.1.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (11.0.0)
Requirement already satisfied: protobuf<6,>=3.20 in /usr/local/lib/python3.10/dist-packages (from streamlit) (4.25.5)
Requirement already satisfied: pyarrow<7.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (17.0.0)
Requirement already satisfied: requests<3,>=2.27 in /usr/local/lib/python3.10/dist-packages (from streamlit) (2.32.3)
Requirement already satisfied: rich<14,>=10.14.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (13.9.4)
Requirement already satisfied: tenacity<10,>=8.1.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (9.0.0)
Requirement already satisfied: toml<2,>=0.10.1 in /usr/local/lib/python3.10/dist-packages (from streamlit) (0.10.2)
Requirement already satisfied: typing-extensions<5,>=4.3.0 in /usr/local/lib/python3.10/dist-packages (from streamlit) (4.12.0)
Collecting watchdog<7,>=2.1.5 (from streamlit)
  Downloading watchdog-6.0.0-py3-none-manylinux2014_x86_64.whl.metadata (44 kB)
44.3/44.3 kB 3.0 MB/s eta 0:00:00
Requirement already satisfied: gitpython!=3.1.19,<4,>=3.0.7 in /usr/local/lib/python3.10/dist-packages (from streamlit) (3.1.19)
Collecting pydeck<1,>=0.8.0b4 (from streamlit)
  Downloading pydeck-0.9.1-py2.py3-none-any.whl.metadata (4.1 kB)
Requirement already satisfied: tornado<7,>=6.0.3 in /usr/local/lib/python3.10/dist-packages (from streamlit) (6.3.3)
Requirement already satisfied: Jinja2 in /usr/local/lib/python3.10/dist-packages (from altair<6,>=4.0->streamlit) (3.1.4)
Requirement already satisfied: jsonschema<3.0 in /usr/local/lib/python3.10/dist-packages (from altair<6,>=4.0->streamlit) (4.17.3)
Requirement already satisfied: narwhals<1.14.2 in /usr/local/lib/python3.10/dist-packages (from altair<6,>=4.0->streamlit) (1.14.2)
Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/lib/python3.10/dist-packages (from gitpython!=3.1.19,<4,>=3.0.7->streamlit) (4.0.1)
Requirement already satisfied: python-dateutil<3,>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas<3,>=1.4.0->streamlit) (2.8.2)
Requirement already satisfied: pytz<2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas<3,>=1.4.0->streamlit) (2020.1)
```

4. Creating the Page

Script :

```

%%writefile app.py
import pandas as pd
import pickle
import streamlit as st

# Load model dan vectorizer
pickle_in = open('classifier.pkl', 'rb')
classifier = pickle.load(pickle_in)

pickle_vectorizer = open('tfidf_vectorizer.pkl', 'rb')
vectorizer = pickle.load(pickle_vectorizer)

# Load data
@st.cache_data
def load_data():
    df = pd.read_csv('/content/streamlit.csv') # Ganti dengan file Anda
    return df

df = load_data()

# Fungsi prediksi
def prediction(title, company, location, salary):
    # Gabungkan input dalam satu string atau array
    user_input = f"{title} {company} {location} {salary}"

    # Transform input menggunakan TF-IDF
    input_vectorized = vectorizer.transform([user_input])

    # Prediksi menggunakan model
    prediction = classifier.predict(input_vectorized)

    # Kembalikan hasil prediksi
    job_map = {0: 'Tidak Layak', 1: 'Layak'}
    job_name = job_map[prediction[0]]
    return job_name

```

```
# Main app
def main():
    st.title('Prediksi Kelayakan Kandidat Kerja')

    # Membuat dropdown dengan variasi unik dari data
    title_options = df['Title'].unique()
    company_options = df['Company'].unique()
    location_options = df['Location'].unique()
    salary_options = df['Salary'].unique()

    # Dropdown untuk input
    title = st.selectbox('Pilih Title', title_options)
    company = st.selectbox('Pilih Company', company_options)
    location = st.selectbox('Pilih Location', location_options)
    salary = st.selectbox('Pilih Salary', salary_options)

    result = ""

    if st.button('Prediksi'):
        result = prediction(title, company, location, salary)
        st.success(result)

if __name__ == '__main__':
    main()
```

Output:

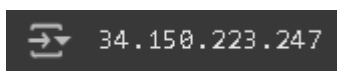


5. Generate the IP

Script :

```
[43] !wget -q -O - ipv4.icanhazip.com
```

Output:



6. Running the Streamlit

Script :

```
[44] !streamlit run app.py & npx localtunnel --port 8501
```

Output:

```
Collecting usage statistics. To deactivate, set browser.gatherUsageStats to false.

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://172.28.0.12:8501
External URL: http://34.150.223.247:8501

your url is: https://yummy-clouds-grab.loca.lt
Stopping...
^C
```

7. Hasil Deployment

Prediksi Kelayakan Kandidat Kerja

Pilih Title

Coal Buyer Seller Specialist (Trader Coal) ▾

Pilih Company

PT Rajawali Bumi Raya ▾

Pilih Location

Manado ▾

Pilih Salary

Rp 4.500.000 ▾

Prediksi

Tidak Layak

BAB III

KESIMPULAN

Berdasarkan hasil evaluasi yang telah dilakukan, skenario **1**, yaitu pelatihan model menggunakan **splitting data 70:30**, menghasilkan nilai akurasi dan ROC AUC yang lebih tinggi dibandingkan dengan skenario lainnya.

Adapun perbandingan model menunjukkan hasil sebagai berikut:

1. **Model SVM** memiliki nilai **akurasi tertinggi**, yaitu sebesar **0.66**.
2. **Model SVM** memiliki **ROC AUC tertinggi**, dengan nilai **0.64**.

Setelah model terbaik diimplementasikan melalui **Streamlit**, sistem mampu melakukan pengklasifikasian kelayakan dengan output berupa “**Layak**” dan “**Tidak Layak**”, berdasarkan atribut **Title**, **Company**, **Location**, dan **Salary**.