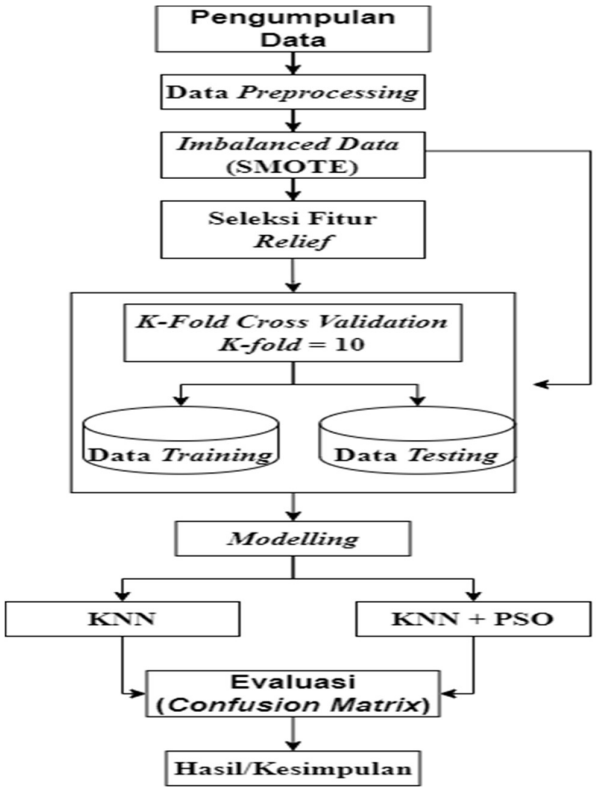
BAB II
METODOLOGI PENELITIAN

2.1. Objek Penelitian

Objek penelitian ini menggunakan data dari BPBD (Badan Penanggulangan Bencana Daerah) dan BMKG (Badan Meteorologi, Klimatologi, dan Geofisika) Kota Samarinda, Dataset ini berisikan data dari tahun 2021 hingga 2023. Adapaun lokasi Penelitian yang dilakukan pada BPBD (Badan Penanggulangan Bencana Daerah) Kota Samarinda yang beralamatkan di JL.Sentosa Dalam No.01, Sungai Pinang Dalam, Kecamatan Sungai Pinang, Kota Samarinda, Kalimantan Timur 75242 dan BMKG (Badan Meteorologi, Klimatologi, dan Geofisika) Kota Samarinda beralamatkan Jl. Pipit No.150, Bandara, Kec. Sungai Pinang, Kota Samarinda, Kalimantan Timur 75117.

2.2. Prosedur Penelitian

Setiap penelitian memilki beberapa langkah dalam tahap pelaksanaan penelitian, Dalam penelitian ini terdapat langkah-langkah untuk mencapai tujuan penelitian. Tahapan penelitian yang dilakukan dimulai dari pengumpulan dan analisis data hingga tahap terakhir yaitu evaluasi hasil. Berikut adalah bagan langkah-langkah yang akan dilakukan :



Gambar 2. 1 *Diagram* Alur Penelitian

2.2.1. Identifikasi Masalah

Identifikasi masalah dalam penelitian merupakan tahap penting yang memungkinkan peneliti untuk menetapkan fokus dan arah penelitian dengan tepat. Permasalahan utama dalam penelitian ini berkaitan dengan penentuan metode optimal untuk mengklasifikasikan data banjir di Kota Samarinda. Selain itu, dilakukan juga tinjauan literatur untuk mengidentifikasi kekosongan penelitian yang terkait dengan klasifikasi data banjir, yang dapat memberikan pandangan mendalam dan bahan yang relevan untuk penelitian ini.

2.2.2. Pengumpulan Data

Tahapan awal pada penelitian ini adalah melakukan pengumpulan data lalu dilanjutkan dengan proses data *preparation*, Penelitian ini menggunakan data Banjir Kota Samarinda dari tahun 2021-2023. Data yang didapatkan dari BPBD (Badan Penanggulangan Bencana Daerah) dan BMKG (Badan Meteorologi, Klimatologi, dan Geofisika) Kota Samarinda, menunjukkan bahwa data tersebut terdiri atas 1095 Dataset dengan 19 Atribut dan 1 Atribut sebagai label didalamnya. Pada dataset Banjir di Kota Samarinda dibentuk tabel yang dapat mempermudah dalam mengetahui berapa banyak fitur yang ada sebagai berikut :

Tabel 2. 1 Fitur *Dataset* Banjir BMKG dan BPBD Kota Samarinda

No	Fitur	Keterangan
1.	Tanggal	Waktu Kejadian
2.	(Tn)	Temperatur-minimum (°C)
3.	(Tx)	Temperatur-maksimum (°C)
4.	(Tavg)	Temperatur-rata-rata (°C)
5.	(RH_avg)	Kelembaban-rata-rata (%)
6.	(RR)	Curah-hujan (mm)
7.	(ss)	Lamanya-penyinaran-matahari (jam)
8.	(ff_x)	Kecepatan-angin-maksimum (m/s)
9	(ff_avg)	Kecepatan-angin-rata-rata (m/s)
10.	(ddd_x)	Arah-angin-maksimum (°)
11.	(ddd_car)	Arah-angin-terbanyak (°)
12.	Jam Kejadian	Jam Terjadinya Bencana
13.	Lokasi Wilayah	Wilayah Terjadinya Bencana
14.	Luas Area M2	Luas area yang terdampak
15.	Objek Terkena Bencana	Fasilitas Yang terdampak bencana
16.	Korban	Jumlah korban Yang terdampak bencana
17.	Kerugian	Nominal Kerugian yang dialami
18.	Keterangan	Detail kejadian bencana
19.	Terjadi Banjir	Ya/Tidak(class)

2.2.3. Data Pre-Procesing

Data yang didapat dari BPBD (Badan Penanggulangan Bencana Daerah) dan BMKG (Badan Meteorologi, Klimatologi, dan Geofisika) harus diolah lebih lanjut sebelum masuk ke tahap pemodelan untuk menghindari pengolahan data yang tidak diperlukan. Tahapan pemrosesan data ini meliputi integrasi data, seleksi, transformasi data, pembersihan data, dan penyeimbangan data sebagai berikut :

a. Data *Integration*

Pada *Data Integration* menggabungkan data dari berbagai sumber yang berbeda menjadi satu set data. Data yang akan digabungkan bersumber dari data Badan Penanggulangan Bencana Daerah (BPBD) dan Badan Meteorologi, Klimatologi, dan Geofisika (BMKG). Data ini mencakup tentang data banjir dan faktor pendukung penyebab banjir Kota Samarinda dari tahun 2021-2023. Proses integrasi data bertujuan untuk memberikan informasi yang lebih lengkap dan akurat terkait data banjir.

b. Data *Selection*

Data selection dilakukan untuk dilakukan pemilihan dan pengambilan data banjir Kota Samarinda yang diperoleh dari BPBD (Badan Penanggulangan Bencana Daerah) dan BMKG (Badan Meteorologi,

Klimatologi, dan Geofisika), proses *data selection* akan difokuskan pada pemilihan data yang relevan dan penting untuk analisis banjir. kemudian untuk atribut yang dianggap kurang relevan maka akan dihapus. Tabel 2.1 merupakan data awal yang diperoleh dari BPBD (Badan Penanggulangan Bencana Daerah) dan BMKG (Badan Meteorologi Klimatologi dan Geofisika) Kota Samarinda, yang terdiri dari 19 kolom. Setelah dilakukan analisis, sebanyak 7 kolom dianggap kurang relevan dan tidak digunakan dalam proses prediksi banjir. Oleh karena itu, dari 19 kolom awal, hanya 11 kolom yang dipilih sebagai fitur atau atribut, sedangkan 1 kolom lainnya dipilih sebagai target atau kelas seperti yang ada pada dalam tabel 2.2.

Tabel 2. 2 *Data Selection*

No	Atribut Awal	Atribut Hasil Seleksi	Keterangan
1	Tgl	Tanggal	Date
2	Tn	Temperatur-minimum	Atribut
3	Tx	Temperatur-maksimum	Atribut
4	Tavg	Temperatur rata-rata	Atribut
5	RH_avg	Kelembaban	Atribut
6	RR	Curah-hujan	Atribut
7	ss	Lamanya-penyinaran-matahari	Atribut
8	ff_x	Kecepatan-angin	Atribut
9	ddd_x	Arah-angin-maksimum	Atribut
10	ff_avg	Kecepatan-angin-rata-rata	Atribut
11	ddd_car	Arah-angin-terbanyak	Atribut
12	Terjadi Banjir	Terjadi-Banjir	Class/target

c. *Data Cleaning*

Data pada data banjir dari BPBD dan BMKG tahun 2021-2023 akan dilakukan serangkaian langkah untuk membersihkan dan mempersiapkan data agar siap digunakan dalam analisis atau pemodelan lebih lanjut. Data *cleaning* sendiri merupakan sebuah proses untuk memperbaiki atau membersihkan data yang tidak tepat, tidak lengkap, atau tidak konsisten. Pada tahap ini, dilakukan pembersihan data dengan menghapus entri yang memiliki nilai #N/A (tidak tersedia) atau data yang kosong, serta mengidentifikasi dan menghapus data yang duplikat.

```
#Penanganan data yang kosong
#Mengecek data kosong
kosong = df.isna().sum().sum()
print(df.isna().sum())
print("Jumlah data kosong: ", kosong)

#Menghitung perbandingan jumlah data sebelum dan sesudah menghapus data kosong
data_kotor = len(df)
data_bersih = df.dropna()

print(f"Jumlah data sebelum pembersihan data kosong: {data_kotor}")
print(f"Jumlah data setelah pembersihan data kosong: {len(data_bersih)}")
```

Gambar 2. 2 Proses *Data Cleaning*

Berikut adalah tabel 2.3 yang berisi tentang tiap parameter yang digunakan dalam kode tersebut beserta penjelasan fungsi tiap parameter.

Tabel 2. 3 *Parameter Data Cleaning*

Parameter	Keterangan
data.isna()	Fungsi ini digunakan untuk melakukan pengecekan, apakah terdapat nilai yang hilang (NaN) pada data.
len()	Fungsi yang digunakan untuk menghitung jumlah elemen dalam suatu objek/data.
dropna()	Fungsi ini digunakan untuk menghilangkan atau menghapus baris atau kolom yang mengandung nilai yang hilang (NaN).

```
# Menghitung jumlah nilai yang hilang di setiap kolom setelah pembersihan
missing_values_after = data_bersih.isnull().sum()
print("Jumlah nilai yang hilang setelah pembersihan:")
print(missing_values_after)
```

Gambar 2. 3 Pemeriksaan nilai yang hilang setelah penghapusan

Pada tabel 2.4 berikut berisi tentang tiap parameter yang digunakan dalam kode tersebut beserta penjelasan fungsi tiap parameter.

Tabel 2. 4 Pemeriksaan nilai yang hilang setelah penghapusan

Parameter	Keterangan
Missing_values_after	Digunakan untuk melakukan tindakan lanjutan setelah tahap penanganan nilai yang hilang selesai.
data.isnull()	Digunakan untuk mengidentifikasi nilai yang hilang dalam sebuah DataFrame atau struktur data lainnya.
sum	Digunakan untuk menghitung jumlah nilai-nilai dalam DataFrame atau Series.

d. Data*Transformation*

Berikut pada Data transformation dilakukan proses pengubahan atau pemrosesan data dari satu bentuk atau format ke bentuk atau format lain yang lebih sesuai atau berguna untuk analisis, pemodelan, atau aplikasi tertentu. Label *encoder* mengacu pada pengubahan nilai label ke dalam bentuk *numeric,* yang ditujukan untuk memudahkan mesin untuk dapat membaca data. Dengan label *encoder*, algoritma *machine learning* dapat dengan dalam mengambil keputusan terbaik dalam mengoperasikan data(Yoga Siswa, 2023). Tujuan lain dari data *transformation* adalah untuk menghasilkan data yang lebih mudah dimengerti, dan sesuai dengan kebutuhan penelitian yang akan dilakukan.

```
#Transformasi data
print(f"===Sebelum Transformasi Data:=== \n{data_bersih[['Arah-angin-terbanyak', 'terjadi-banjir']].head()} \n")#Proses Encoding
```

Gambar 2. 4 Sebelum *Transformasi Data*

Pada Gambar 2.5 merupakan Proses *Encoding*

```
#Transformasi data
print(f"===Sebelum Transformasi Data:=== \n{data_bersih[['Arah-angin-terbanyak', 'terjadi-banjir']].head()} \n")#Proses Encoding
ordinal = OrdinalEncoder()
labelEncoder = LabelEncoder()
data_transform = data_bersih['Arah-angin-terbanyak']

data_bersih['Arah-angin-terbanyak'] = labelEncoder.fit_transform(data_bersih['Arah-angin-terbanyak'])
data_bersih['terjadi-banjir'] = data_bersih['terjadi-banjir'].replace({'banjir':1, 'tidak banjir':0})

print(f"\n===Setelah Transformasi Data:=== \n{data_bersih[['Arah-angin-terbanyak', 'terjadi-banjir']].head()}")
```

Gambar 2. 5 Proses *Encoding*

Berikut adalah tabel 2.5 yang berisi tentang tiap parameter yang digunakan dalam kode tersebut beserta penjelasan fungsi tiap parameter.

Tabel 2. 5 *Parameter Data Transformation*

Parameter	Keterangan
OrdinalEncoder	Berfungsi mengubah fitur kategorikal menjadi representasi numerik
LabelEncoder()	Berguna untuk melakukan pengkodean label pada data.
Data_transform	Berfungsi menyimpan fitur atau atribut dari dataset yang diubah menjadi representasi *numerik.*
Fit_transfrom	menyesuaikan *encoder* dengan data dan langsung mengubah data menjadi nilai numerik.
Replace	metode Pandas yang digunakan untuk menggantikan nilai spesifik dalam DataFrame.
Head	digunakan untuk menampilkan beberapa baris pertama dari DataFrame tersebut

e. Data *Balancing*

Data *balancing* merupakan proses yang dilakukan untuk menyeimbangkan distribusi kelas atau label pada dataset. Hal ini sering kali diperlukan dalam konteks masalah klasifikasi di mana terdapat ketidakseimbangan yang signifikan antara jumlah sampel yang termasuk dalam setiap kelas atau label. Pada penelitian ini menggunakan metode *Synthetic Minority Over-sampling Technique* (SMOTE) dalam menyeimbangkan data yang tidak seimbang (*imbalance data*). Metode SMOTE digunakan untuk menghasilkan sampel sintetis dari kelas minoritas, sehingga meningkatkan representasi kelas minoritas dalam dataset.

```
#Memisahkan variabel atribut dan target
X = data_bersih.drop(['Tanggal', 'terjadi-banjir'], axis=1)
y = data_bersih['terjadi-banjir']

#Jumlah data pada variable kelas sebelum diterapkan oversampling SMOTE
before = y.value_counts()

#Penerapan oversampling SMOTE
smote = SMOTE(random_state=42)
X_res, y_res = smote.fit_resample(X, y)

#Jumlah data pada variable kelas setelah diterapkan oversampling SMOTE
after = y_res.value_counts()

#Visualisasi perbandingan sebelum dan sesudah diterapkan oversampling SMOTE
fig, ax = plt.subplots(1, 2, figsize=(12, 6))

ax[0].bar(before.index, before.values)
ax[0].set_title('Sebelum SMOTE')
ax[0].set_xlabel('Kelas')
ax[0].set_ylabel('Jumlah')

ax[1].bar(after.index, after.values)
ax[1].set_title('Setelah SMOTE')
ax[1].set_xlabel('Kelas')
ax[1].set_ylabel('Jumlah')

plt.show()
```

Gambar 2. 6 *Data Balancing*

Berikut adalah tabel 2.6 yang berisi tentang tiap parameter yang digunakan dalam kode tersebut beserta penjelasan fungsi tiap parameter.

Tabel 2. 6 *Parameter Data Balancing*

Parameter	Keterangan
drop()	Fungsi drop digunakan untuk melakukan penghapusan terhadap kolom atau atribut yang diinginkan.
value_counts()	Fungsi ini digunakan untuk menghitung jumlah setiap nilai pada atribut yang diinginkan.
SMOTE()	Untuk membuat sebuah objek SMOTE yang digunakan untuk *oversampling.*
fit_resample	Metode yang digunakan oleh objek SMOTE yang telah dibuat sebelumnya untuk menerapkan teknik *oversampling* pada data
plt.subplots	Membuat grid subplot dengan 1 baris dan 2 kolom
figsize	Menentukan ukuran dari keseluruhan figur. Lebar 12 inci dan tinggi 6 inci.
Plt.show	Menampilkan plot yang telah dibuat

2.2.4. Pembagian Data

Dalam penelitian klasifikasi menggunakan algoritma KNN, Selanjutnya dataset yang akan digunakan dibagi menjadi dua bagian utama: data latih dan data uji. Untuk memastikan evaluasi model yang akurat dan konsisten, peneliti mengadopsi teknik *K-Fold Cross Validation*. pengujian menggunakan *k-fold cross validation*, dengan seiring bertambahnya nilai K (tetangga terdekat) maka akan mempengaruhi nilai akurasi (Nabila et al., 2021). Teknik ini diimplementasikan melalui *library sklearn.model_selection* dengan menggunakan fungsi *cross_val_score* di lingkungan Python. Dengan pendekatan ini, peneliti dapat memvalidasi kinerja model secara menyeluruh dengan membagi dataset

dan menghitung skor rata-rata dari hasil pengujian. Teknik *K-Fold Cross Validation* yang digunakan dengan membagi data menjadi 10 kelompok, yang berarti data akan dibagi menjadi 10 bagian.

```
# K-fold cross validation
kf = KFold(n_splits=10, shuffle=True, random_state=42)

for k in k_values:
    print(f"\nk = {k}")

    fold_scores = []
    fold_confusion_matrices = []
    kFold = 1

    for train_index, test_index in kf.split(X_res, y_res):
        X_train, X_test = X_res.iloc[train_index], X_res.iloc[test_index]
        y_train, y_test = y_res.iloc[train_index], y_res.iloc[test_index]
```

Gambar 2. 7 Pembagian Data

Berikut adalah tabel 2.7 yang berisi tentang tiap parameter yang digunakan dalam kode tersebut beserta penjelasan fungsi tiap parameter.

Tabel 2. 7 *Parameter* Pembagian Data

Parameter	Keterangan
KFold	Metode validasi silang untuk membagi data menjadi beberapa lipatan yang kemudian akan menggunakanya secara bergantian sebagai *dataset training* dan *testing*.
n_splits=5	Menentukan jumlah lipatan yang akan dibuat.
shuffle=True	Menginisiasikan apakah ingin menyatukan kembali data sebelum membaginya menjadi lipatan atau tidak.
random_state=42	Nilai yang digunakan untuk melakukan kontrol terhadap proses penyatuan kembali agar hasil dapat direproduksi secara konsisten
Train_index & *test_index*	Setiap iterasi dari loop yang terjadi akan mendapatkan indeks yang dihasilkan untuk data *training* dan *testing*.
kf.split(X)	Metode yang mengembalikan generator yang menghasilkan indeks untuk setiap lipatan, yang dimana setiap iterasi akan menghasilkan indeks untuk dataset *training* dan *testing*.
X.iloc[train_index], X.iloc[test_index]	Menggunakan indeks yang telah dihasilkan untuk memilih baris yang sesuai dari atribut yang ada pada 'X' untuk dataset *training* dan *testing*.
y.iloc[train_index], y.iloc[test_index]	Menggunakan indeks yang telah dihasilkan untuk memilih baris yang sesuai dari atribut yang ada pada 'y' untuk dataset *training* dan *testing*.

2.2.5. Modelling

Pada tahap ini, akan diuraikan mengenai model yang digunakan dalam penelitian ini. Model klasifikasi yang digunakan adalah *K-Nearest Neighbor* (KNN) dengan *Relief* sebagai teknik seleksi fitur dan *Particle Swarm Optimization* (PSO) sebagai metode optimasi. Sebelum dataset diproses, pendekatan *Synthetic Minority Oversampling Technique* (SMOTE) akan digunakan untuk menangani ketidak seimbangan data(*Imbalanced data*). akhir dari penelitian akan melakukan perbandingan dengan menggunakan seleksi fitur *Relief* dan tanpa menggunakan seleksi fitur *Relief*. berikut proses pemodelannya.

a. Permodelan Algoritma KNN

Tahap awal Menyiapkan dataset banjir yang telah melalui tahap data balancing menggunakan metode *oversampling* SMOTE dengan detailnya seperti yang dijelaskan pada gambar 2.5 lalu dibagi menjadi data *training* dan data *testing* yang dikhususkan untuk model dalam mempelajari pola data dengan menggunakan teknik *10-Fold Cross-Validation.*

Selanjutnya permodelan ini akan menggunakan *K-Nearest Neighbor* (KNN) dalam mencari hasil evaluasi *confusion matrix* dan juga prediksi nilai akurasi.

```
# Inisiasi algoritma k-nearest neighbors
knn = KNeighborsClassifier(n_neighbors=k, metric='euclidean')

# Melatih model KNN
knn.fit(X_train, y_train)

# Predict on the test set
y_pred = knn.predict(X_test)
```

Gambar 2. 8 Persiapan *Insialisasi Model KNN*

Pada tabel 2.8 berisikan tentang tiap parameter yang digunakan dalam kode tersebut beserta penjelasan fungsi tiap parameter.

Tabel 2. 8 *Parameter* Persiapan *Insialisasi Model KNN*

Parameter	Keterangan
KneighborsClassifier()	kelas dari library scikit-learn yang digunakan untuk membuat model klasifikasi menggunakan algoritma *K-Nearest Neighbors* (KNN).
n_neighbors=5	parameter yang ditentukan saat menginstansiasi *KNeighborsClassifier,* Parameter ini menetapkan jumlah tetangga terdekat yang akan digunakan dalam algoritma KNN.
knn.fit()	metode yang digunakan untuk melatih model KNN pada data pelatihan.
X_train	variabel yang menyimpan data fitur yang digunakan untuk melatih model.
y_train	variabel yang menyimpan label atau target yang sesuai dengan data fitur di *X_train.*
predict()	metode yang digunakan untuk membuat prediksi menggunakan model KNN yang telah dilatih.

Adapun rumus yang biasa digunakan dalam melakukan perhitungan algoritma *K-Nearest Neighbor* (KNN) adalah sebagai berikut :

$$\sqrt{\sum_{i=1}^{p} \quad (\alpha_{k-} b_k)^2}$$

(2. 1)

Sumber : (Jatmiko Indriyanto, 2021)

Keterangan :

α_K : Sampel data

b_k : Data uji atau Data testing

P : Dimensi data

i : variable data

b. Permodelan Algoritma KNN – PSO

Pemodelan ini akan menerapkan *Particle Swarm Optimization* (PSO) untuk mengoptimalkan parameter-parameter algoritma klasifikasi dengan tujuan meningkatkan performa keseluruhan model. Tahapan dalam pemodelan ini dimulai dengan menginstal pustaka *pyswarm* Berikut adalah penerapan PSO pada model yang akan dioptimalkan.

```
!pip install pyswarm
```

Gambar 2. 9 *Menginstall PSO*

Setelah melakukan pengginstalan, selanjutnya dilakukan pengimporan modul PSO dari *pyswarm* dan penerapan PSO

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score
from pyswarm import pso  # Make sure you have the PSO library installed

# Function to optimize KNN
def optimize_knn(x, k):
    leaf_size = int(x[0])
    p = int(x[1])

    clf = KNeighborsClassifier(n_neighbors=k, leaf_size=leaf_size, p=p)
    scores = cross_val_score(clf, X_res, y_res, cv=10)

    return -scores.mean()  # Maximizing accuracy by minimizing the negative score

# Bounds for KNN hyperparameters
lb = [10, 1]  # Lower bounds for leaf_size and p
ub = [50, 2]  # Upper bounds for leaf_size and p

# List of odd k values to optimize
k_values = [3, 5, 7, 9, 11, 13, 15]

optimal_parameters = {}

for k in k_values:
    # Perform PSO optimization for each k
    xopt, fopt = pso(optimize_knn, lb, ub, args=(k,), swarmsize=50, maxiter=30)
```

Gambar 2. 10 *Mengimpor* dan *menerapkan PSO* Pada *Model*

Berikut adalah tabel 2.9 yang berisi tentang tiap parameter yang digunakan dalam kode tersebut beserta penjelasan fungsi tiap parameter.

Tabel 2. 9 Parameter *Mengimpor* dan *menerapkan PSO* Pada *Model*

Parameter	Keterangan
optimize_knn(x)	Sebuah fungsi objektif yang akan dioptimalkan yang menerima vector *'x'* berisikan parameter yang akan dioptimalkan.
cross_val_score()	Menginisialisasi model *K-Nearest Neighbort* berdasarkan parameter yang diberikan. Dalam fungsi tersebut melakukan *cross validation* dengan *5-fold cross-validation*. Kemudian mengembalikan rata-rata negative dari skor *cross validation*.
lb	Batas bawah untuk setiap parameter dalam vektor *'x'*.
ub	Batas atas untuk setiap parameter dalam vektor *'x'*.

xopt, fopt	Berfungsi untuk mendapatkan kombinasi parameter terbaik yang meminimalkan fungsi *'optimize_rf'*.
swarmsize=50	Mengindikasikan bahwa terdapat 50 partikel dalam *swarm* yang akan dievaluasi pada setiap iterasi.
maxiter=30	Jumlah iterasi maksimal yang akan dijalankan oleh PSO.

c. Permodelan Algoritma KNN –RELIEF

Dalam penelitian ini, algoritma *Relief* akan digunakan untuk seleksi fitur. Pada tahap seleksi fitur, fitur-fitur yang ada akan diurutkan berdasarkan pengaruhnya terhadap hasil prediksi, dimulai dari fitur yang memiliki pengaruh terbesar hingga fitur yang memiliki pengaruh terkecil atau bahkan tidak berpengaruh sama sekali. Tahapannya meliputi mengimpor modul *Relief* dari library *sklearn* dan modul pandas. Modul pandas digunakan untuk mengubah dataset menjadi bentuk dataframe. Setelah itu, fitur-fitur akan dirangking berdasarkan skor yang diperoleh dari algoritma *Relief*, berikut penerapannya.

```
!pip install ReliefF
```

Gambar 2. 11 *Menginstall Relief*

Berikut adalah tabel 2.10 yang berisi tentang tiap parameter yang digunakan dalam kode tersebut beserta penjelasan fungsi tiap parameter.

Tabel 2. 10 Parameter *Relief*

Parameter	Keterangan
SelectKBest()	Fungsi dari *library sklearn* yang digunakan untuk memilih fitur-fitur terbaik berdasarkan nilai kriteria tertentu.
f_classif	Fungsi yang digunakan untuk mendapatkan skor relief yang akan digunakan sebagai kriteria untuk pemilihan fitur.
fit_transform	Kombinasi dari fit dan transform. Pertama, fit menyesuaikan selektor dengan data, kemudian transform menerapkan seleksi fitur pada data.
X_res	Artibut hasil dari proses *oversampling* SMOTE.
Y_rres	Label atau target yang telah diproses dala
*Fs.top_features_*	Mendapatkan indeks fitur yang dipilih berdasarkan skor tertinggi.
selected_features_names	Menyimpan nama dari fitur yang dipilih.
pd.DataFrame	Digunakan untuk membuat *dataframe* berdasarkan skor dan *p-value* yang dihasilkan sebelumnya.
feature_weights	Menyimpan bobot atau skor pentingnya dari fitur-fitur yang dipilih berdasarkan hasil perhitungan algoritma *Relief*.
*fs.feature_importances_*	Atribut ini berisi bobot atau skor pentingnya setiap fitur yang dihitung oleh algoritma ReliefF.
selected_features_indices	Indeks fitur yang dipilih.

Adapun rumus Langkah-langkah untuk melakukan seleksi fitur *Relief* sebagai berikut:

a. Inisialisasi nilai awal seluruh bobot fitur = 0 dan menentukanjumlah iterasi.
b. Memilih sebuah data yang akan dijadikan sebagai titik acak atau titik pusat.
c. Mencari miss dan hitterdekat dengan cara menghitung jarak antara titik pusat dengan data yang memiliki kelas yang sama. Jarak terdekat antara titik pusat dan data pada kelas

positif disebut hit.Sedangkan, jarak terdekat antara titik pusat dengan data yang pada kelas negatifdisebut miss.

d. Lakukan update bobot untuk setiap fitur. Fitur dengan data kategori dihitung menggunakan Persamaan 2.2.

$$diff(A, Ri, HM) = \{0; value(A, Ri) = value(A, HM) \\ 1; otherwise \quad (2.2)$$

e. Sedangkan, fitur dengan data numerik dihitung menggunakan persamaan 2.3.

$$diff(A, Ri, HM) = \frac{\text{value(A,Ri)} - \text{valu (A,HM)}|}{\max(A) - \text{m} \quad (A)} \quad (2.3)$$

f. Sehingga rumus perbaruan bobot dihitung menggunakan persamaan 2.4

$$W[A] = W[A] - diff(A, Ri, H)m + diff(A, Ri, M)m \quad (2.4)$$

Selanjutnya,dilanjutkan dengan iterasi selanjutnya yang dimulai dari langkah 1 hingga bobot fitur yang baru telah didapat.

c. Permodelan Algoritma KNN - RELIEF –PSO

Dalam penelitian ini, akan mengkombinasi tiga algoritma yang masing-masing telah diselesaikan tahapannya, algoritma tersebut yakni KNN, *Relief*, dan PSO. Tahap awal dimulai dengan mengimpor pustaka, termasuk dari pustaka *sklearn*, modul *SelectKBes*t dan *Relief* dari modul seleksi fitur untuk memilih fitur, serta *KNeighborsClassifier* untuk membuat model KNN, dan modul PSO dari *library pyswarm* untuk mengoptimalkan PSO. Selanjutnya, akan dilakukan proses persiapan dan pembagian data menggunakan *kfold cross validation*, diikuti dengan penerapan algoritma seleksi fitur untuk mendapatkan fitur yang relevan. Langkah berikutnya adalah menerapkan optimasi menggunakan PSO untuk menyesuaikan parameter-parameter algoritma klasifikasi dengan tujuan meningkatkan performa keseluruhan model. Setelah itu, kami melatih model klasifikasi KNN dengan parameter yang telah dioptimalkan.

d. Perbandingan Hasil

Hasil akhirnya akan dilakukan perbandingan mengenai penerapan *Feature Selection* dan tanpa *Feature Selection* untuk melihat seberapa optimal *KNN* bekerja dalam menemukan hasil optimal baik dengan menggunakan *Oversampling* dan optimasi maupun tidak sama sekali.

2.2.5. Evaluasi

Dalam penelitian ini, kinerja model KNN akan dievaluasi dengan membandingkan beberapa model KNN sebelumnya, baik yang menggunakan teknik seleksi fitur maupun yang tidak. Evaluasi model dilakukan untuk mendapatkan model terbaik melalui pengujian terhadap data uji berdasarkan *Confusion Matrix.* Evaluasi dilakukan menggunakan accuracy sebagai metrik, yang dapat dihitung dengan persamaan berikut :

$$Accuracy = \frac{\text{TP + TN}}{\text{TP + FP + TN + FN}} X100\% \quad (2.5)$$

Keterangan:

TP (*True Positive*) : jumlah data yang berlabel yes diklasifikasikan sebagai benar oleh model.

TN (*True Negative*) : jumlah data yang berlabel no diklasifikasikan sebagai salah oleh model.

FP *(False Positive)* : jumlah data yang berlabel yes seharusnya salah.

FN (*False Negative*) : jumlah data yang berlabel no diklasifikasikan padahal seharusnya benar.