Laporan Tugas Besar 2

IF3070 Dasar Inteligensi Artifisial

Implementasi Algoritma Pembelajaran Mesin

Disusun Oleh:

Kelompok 1 K03

Muhammad Daffa Kusuma	/ 18222108
Ahmad Fawwazi	/ 18222117
Muhammad Faishal Putra	/ 18222129
Muhammad Faishal Firdaus	/ 18222136

Program Studi Sistem dan Teknologi Informasi

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

2024

1. Implementasi KNN

1.1. Implementasi KNN from *scratch*

Berikut merupakan potongan code implementasi KNN from *scratch*:

```
from math import sqrt

# Fungsi untuk menghitung jarak Euclidean
def euclidean_distance(row1, row2):
    # Validasi panjang data sekali saja
    assert len(row1) == len(row2)
        distance = sum((row1[i] - row2[i]) ** 2 for i in
range(len(row1)))
    return sqrt(distance)

# Fungsi untuk mendapatkan K tetangga terdekat
def get_neighbors(dataset, row, num_neighbors):
    distances = []
    for set_row in dataset:
        dist = euclidean_distance(row, set_row)
        distances.append((set_row, dist))
    distances.sort(key=lambda x: x[1])
            neighbors = [distances[i][0] for i in
range(num_neighbors)]
    return neighbors

# Fungsi untuk memprediksi kelas berdasarkan tetangga
terdekat
def predict_classification(train, row, num_neighbors):
    neighbors = get_neighbors(train, row, num_neighbors)
    labels = [neighbor[-1] for neighbor in neighbors]
    prediction = max(set(labels), key=labels.count)
    return prediction

# Fungsi untuk menghitung akurasi
def accuracy_metric(actual, predicted):
    correct = sum(1 for i in range(len(actual)) if actual[i]
== predicted[i])
    return (correct / len(actual)) * 100

# Fungsi untuk memulai proses KNN
def knn_classification(dataset, x_set, y_set,
num_neighbors):
    # Sinkronisasi kolom antara train dan validasi
```

```
                                                common_cols =
dataset.columns.intersection(x_set.columns)
          dataset = dataset[common_cols]
          x_set = x_set[common_cols]

            # Mengonversi dataframe ke list setelah mengisi nilai
kosong dan memastikan numerik
          dataset = dataset.values.tolist()
          x_set = x_set.values.tolist()
          y_set = y_set.values.tolist()

          predictions = []
          for row in x_set:
                    result = predict_classification(dataset, row,
num_neighbors)
              predictions.append(result)

          # Menghitung akurasi
             accuracy = accuracy_metric(y_set, predictions)    #
Sinkronkan dengan jumlah baris uji

          # Menampilkan hasil akurasi
          return accuracy

      knn_classification(train_sets, x_trains, y_trains, 5)
```

Implementasi algoritma K-Nearest Neighbors (KNN)(*from scratch*) dalam kode di atas mencakup beberapa langkah. Pertama, fungsi euclidean_distance digunakan untuk menghitung jarak Euclidean antara dua titik data, memastikan bahwa jarak dihitung secara akurat untuk semua fitur numerik. Kemudian, fungsi get_neighbors menemukan tetangga terdekat berdasarkan jarak terpendek dalam dataset. Setelah itu, fungsi predict_classification memprediksi kelas data uji dengan mengambil kelas mayoritas dari tetangga terdekat yang ditemukan.

Selanjutnya, fungsi accuracy_metric digunakan untuk mengevaluasi performa model dengan membandingkan prediksi model dengan label sebenarnya. Fungsi inti, knn_classification, bertugas untuk menggabungkan langkah-langkah ini: mengonversi dataset menjadi format yang sesuai,

melakukan prediksi untuk setiap baris data uji, dan akhirnya menghitung akurasi model. Dalam proses ini, data pelatihan dan validasi harus diproses dengan baik, termasuk menghapus nilai kosong, memastikan kolom fitur dan label sinkron, serta memastikan semua data berada dalam format numerik. Kode ini dirancang untuk fleksibilitas, sehingga pengguna dapat menyesuaikan jumlah tetangga (num_neighbors) untuk eksperimen dan analisis.

1.2. Implementasi KNN menggunakan *scikit-learn*

Berikut merupakan potongan code implementasi KNN menggunakan *scikit-learn*:

```
from math import sqrt
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Fungsi untuk memulai proses KNN dengan sklearn
def knn_classification_sklearn(x_train, y_train, x_test,
y_test, num_neighbors):
    # Membuat model KNN
    model = KNeighborsClassifier(n_neighbors=num_neighbors)
    model.fit(x_train, y_train)
    predictions = model.predict(x_test)
    # Menghitung akurasi
    accuracy = accuracy_score(y_test, predictions) * 100
    return accuracy

# Menjalankan KNN dengan sklearn
knn_classification_sklearn(x_trains, y_trains, x_vals,
y_vals , 5)
```

Implementasi K-Nearest Neighbors (KNN) menggunakan *scikit-learn* mempermudah proses dengan efisiensi tinggi. Model KNN dibuat menggunakan KNeighborsClassifier dengan parameter n_neighbors untuk menentukan jumlah tetangga terdekat. Data pelatihan dilatih menggunakan .fit(), lalu prediksi dilakukan pada data uji menggunakan .predict(). Akurasi dihitung dengan membandingkan hasil prediksi dan label sebenarnya menggunakan accuracy_score. Pendekatan ini mengotomatiskan perhitungan jarak dan pemilihan tetangga, sehingga cocok untuk dataset besar.

2. Implementasi Naive-Bayes

2.1. Implementasi Naive-Bayes menggunakan *scikit-learn*

Berikut merupakan potongan codeNaive-Bayes menggunakan *scikit-learn*:

```
from math import sqrt
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

# Fungsi untuk memulai proses Naive Bayes dengan sklearn
def naive_bayes_classification_sklearn(x_train, y_train,
x_test, y_test):
    # Membuat model Naive Bayes
    model = GaussianNB()
    model.fit(x_train, y_train)
    predictions = model.predict(x_test)
    # Menghitung akurasi
    accuracy = accuracy_score(y_test, predictions) * 100
    return accuracy


# Menjalankan Naive Bayes dengan sklearn
naive_bayes_classification_sklearn(x_trains, y_trains,
x_vals, y_vals)
```

Implementasi Naive Bayes menggunakan *scikit-learn* adalah proses yang sederhana dan efisien. Dalam kode di atas, algoritma Naive Bayes direalisasikan dengan menggunakan GaussianNB, yang dirancang untuk bekerja dengan data kontinu berdasarkan asumsi distribusi normal (Gaussian). Proses ini dimulai dengan membuat model Naive Bayes menggunakan kelas GaussianNB dari *scikit-learn*. Model ini kemudian dilatih pada data pelatihan (x_train dan y_train) menggunakan fungsi .fit(). Setelah model dilatih, prediksi dilakukan pada data uji (x_test) dengan menggunakan fungsi .predict(). Hasil prediksi dibandingkan dengan label asli dari data uji (y_test) untuk menghitung akurasi model menggunakan fungsi accuracy_score. Akurasi dinyatakan dalam persentase untuk memberikan gambaran kinerja model secara keseluruhan. Pendekatan ini sangat cocok untuk data dengan distribusi fitur yang sesuai dengan asumsi Naive Bayes, seperti independensi antar fitur.

3. *Cleaning* dan *Preprocessing*

Tahap cleaning dan preprocessing data dilakukan untuk memastikan kualitas data yang digunakan pada model pembelajaran mesin, baik KNN maupun Gaussian Naive Bayes. Berikut adalah langkah-langkah yang dilakukan pada tahap cleaning:

1. Handling Missing Data

Berikut merupakan potongan code implementasi Handling Missing Data:

```
mean_data                                                    =
['DomainLength','CharContinuationRate','URLCharProb','TLDLength',
'DegitRatioInURL']
median_data                                                  =
['LetterRatioInURL','DomainTitleMatchScore','URLTitleMatchScore',
'NoOfiFrame','URLLength','TLDLegitimateProb','NoOfLettersInURL','
LineOfCode','LargestLineLength','NoOfImage','NoOfCSS','NoOfJS','N
oOfSelfRef','NoOfEmptyRef','NoOfExternalRef','NoOfOtherSpecialCha
rsInURL','NoOfDegitsInURL','NoOfEqualsInURL','NoOfQMarkInURL']
modus_data                                                   =
['Robots','HasTitle','SpacialCharRatioInURL','NoOfAmpersandInURL'
,'HasObfuscation','IsDomainIP','NoOfSubDomain','NoOfObfuscatedCha
r','ObfuscationRatio','HasTitle','NoOfURLRedirect','NoOfSelfRedir
ect','NoOfPopup','HasExternalFormSubmit','HasPasswordField','Bank
','Crypto','Pay']
domain_know                    =                    ['HasFavicon',
'HasDescription','IsHTTPS','HasFavicon','IsResponsive','HasDescri
ption','HasSocialNet','HasSubmitButton','HasHiddenFields','HasCop
yrightInfo']
for col in domain_know:
           train_set.loc[(train_set['label']      ==      1)      &
(train_set[col].isnull()), col] = 1
   train_set.loc[(df['label'] == 0) & (train_set[col].isnull()),
col] = 0
# Menggunakan SimpleImputer untuk mengisi nilai kosong dengan
mean
imputer = SimpleImputer(strategy='mean')
train_set[mean_data]                                         =
imputer.fit_transform(train_set[mean_data])

# Menggunakan SimpleImputer untuk mengisi nilai kosong dengan
median
```

```
imputer2 = SimpleImputer(strategy='median')
train_set[median_data]                                          =
imputer2.fit_transform(train_set[median_data])

imputer3 = SimpleImputer(strategy='most_frequent')

# Mengisi missing values dengan modus
train_set[modus_data]                                           =
imputer3.fit_transform(train_set[modus_data])

sumMissingValues = train_set.isnull().sum()
sumMissingValues
train_set.head()
```

Langkah pertama dilakukan dengan mengelompokkan kolom berdasarkan strategi penanganan missing data. Untuk kolom numerik seperti DomainLength, CharContinuationRate, dan URLCharProb, nilai kosong diisi dengan rata-rata (mean) karena data pada kolom-kolom ini diasumsikan memiliki distribusi normal sehingga rata-rata dapat merepresentasikan nilai tengah dengan baik. Sementara itu, untuk kolom numerik lain seperti URLLength dan LineOfCode yang berpotensi memiliki distribusi data yang skewed atau outlier signifikan, nilai kosong diisi dengan median untuk menghindari pengaruh outlier pada data. Selain itu, kolom kategorikal seperti Robots, HasTitle, dan IsDomainIP diisi dengan modus, yaitu nilai yang paling sering muncul, karena nilai dominan dianggap mampu merepresentasikan kategori umum dalam data.

Penanganan khusus untuk kolom yang terkait dengan domain knowledge, seperti HasFavicon dan IsHTTPS. Pada kolom-kolom ini, nilai kosong diisi berdasarkan label URL. Jika label menunjukkan URL legitimate (1), nilai kosong diisi dengan 1, dan jika label menunjukkan phishing (0), nilai kosong diisi dengan 0.Hal ini dilakukan untuk menghasilkan nilai yang relevan dengan karakteristik data. Pengisian dilakukan menggunakan library SimpleImputer dari scikit-learn.

2. Dealing with Outliers

Berikut merupakan potongan code implementasi Handling Missing Data:

```
for col in mean_data:
    q1 = train_set[col].quantile(0.25)
```

```
    q3 = train_set[col].quantile(0.75)
    IQR = q3 - q1
    lower_bound = q1 - 1.5 * IQR
    upper_bound = q3+ 1.5 * IQR
    train_set[col] = np.where(
            (train_set[col] < lower_bound) | (train_set[col] >
upper_bound),
        train_set[col].mean(),  # Mengganti outlier dengan mean
        train_set[col]
    )

for col in median_data:
    q1 = train_set[col].quantile(0.25)
    q3 = train_set[col].quantile(0.75)
    IQR = q3 - q1
    lower_bound = q1 - 1.5 * IQR
    upper_bound = q3 + 1.5 * IQR
    train_set[col] = np.where(
            (train_set[col] < lower_bound) | (train_set[col] >
upper_bound),
        train_set[col].median(),  # Mengganti outlier dengan mean
        train_set[col]
    )

for col in modus_data:
    q1 = train_set[col].quantile(0.25)
    q3 = train_set[col].quantile(0.75)
    IQR = q3 - q1
    lower_bound = q1 - 1.5 * IQR
    upper_bound = q3+ 1.5 * IQR
    train_set[col] = np.where(
            (train_set[col] < lower_bound) | (train_set[col] >
upper_bound),
        train_set[col].mode(),
        train_set[col]
    )
```

Proses deteksi outlier pada semua kolom dilakukan dengan menghitung Q1 (kuartil pertama) dan Q3 (kuartil ketiga), lalu menghitung IQR sebagai:

$$Q3 - Q1$$

Batas bawah dan atas didefinisikan masing-masing sebagai:

$$Q1 - 1.5 \times IQR \text{ dan } Q3 + 1.5 \times IQR.$$

Nilai yang berada di luar batas ini dianggap sebagai outlier dan digantikan sesuai dengan strategi yang telah ditentukan. Dengan cara ini dapat memastikan nilai ekstrem yang berpotensi mengganggu analisis atau performa model dihilangkan tanpa mengubah distribusi data secara signifikan.

```
from sklearn.base import BaseEstimator, TransformerMixin
import numpy as np
import pandas as pd

class ClusterOutlier(BaseEstimator, TransformerMixin):
          def   __init__(self,   mean_cols=None,   median_cols=None,
modus_cols=None):

        self.mean_cols = mean_cols
        self.median_cols = median_cols
        self.modus_cols = modus_cols

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        X = X.copy()


        for col in self.mean_cols:
            q1 = X[col].quantile(0.25)
            q3 = X[col].quantile(0.75)
            IQR = q3 - q1
            lower_bound = q1 - 1.5 * IQR
            upper_bound = q3 + 1.5 * IQR
            X[col] = np.where(
                (X[col] < lower_bound) | (X[col] > upper_bound),
                X[col].mean(),
                X[col]
            )
```

```
        for col in self.median_cols:
            q1 = X[col].quantile(0.25)
            q3 = X[col].quantile(0.75)
            IQR = q3 - q1
            lower_bound = q1 - 1.5 * IQR
            upper_bound = q3 + 1.5 * IQR
            X[col] = np.where(
                (X[col] < lower_bound) | (X[col] > upper_bound),
                X[col].median(),
                X[col]
            )

        for col in self.modus_cols:
            q1 = X[col].quantile(0.25)
            q3 = X[col].quantile(0.75)
            IQR = q3 - q1
            lower_bound = q1 - 1.5 * IQR
            upper_bound = q3 + 1.5 * IQR
                        modus_value = X[col].mode()[0] if not
X[col].mode().empty else np.nan
            X[col] = np.where(
                (X[col] < lower_bound) | (X[col] > upper_bound),
                modus_value,
                X[col]
            )

        return X
coba = df.copy()

outlier_handler = ClusterOutlier(mean_cols=mean_data,
median_cols=median_data, modus_cols=modus_data)
train_set_cleaned = outlier_handler.fit_transform(coba)
```

Class ClusterOutlier menangani outlier dalam dataset dengan tiga strategi utama: penggantian menggunakan mean, median, dan modus. Class ini dirancang untuk fleksibilitas, dimana pengguna dapat menentukan kolom mana yang diproses dengan masing-masing strategi melalui parameter mean_cols, median_cols, dan modus_cols. Dalam metode transform, outlier dideteksi menggunakan metode Interquartile Range (IQR). Setelah class diinisialisasi, dataset dapat diproses langsung

menggunakan metode fit_transform, menghasilkan data yang bersih dari outlier tanpa mengubah karakteristik asli data. Cara ini dapat memastikan efisiensi, konsistensi, dan kualitas data untuk analisis atau pelatihan model pembelajaran mesin.

```
def calculate_outliers(set):
    # Menghitung Q1, Q3, dan IQR untuk semua kolom numerik
    q1 = set.quantile(0.25)
    q3 = set.quantile(0.75)
    IQR = q3 - q1

    # Mendefinisikan batas bawah dan atas untuk mendeteksi
outlier
    lower_bound = q1 - 1.5 * IQR
    upper_bound = q3 + 1.5 * IQR

    # Menandai outlier pada setiap kolom
    outliers = ((set < lower_bound) | (set > upper_bound))

    # Menghitung jumlah outlier per kolom
    outlier_counts = outliers.sum()

    return outlier_counts

# Pastikan `train_set` adalah DataFrame yang sudah difilter hanya
untuk kolom numerik
numeric_cols = train_set.select_dtypes(include=['float'])
outlier_counts = calculate_outliers(numeric_cols)

print(train_set_cleaned)
```

calculate_outliers digunakan untuk mendeteksi dan menghitung jumlah outlier dalam dataset berdasarkan kolom numerik. Fungsi ini dimulai dengan menghitung Q1 (kuartil pertama) dan Q3 (kuartil ketiga) untuk setiap kolom numerik, lalu menentukan Interquartile Range (IQR) sebagai selisih antara Q3 dan Q1. Fungsi ini kemudian menghasilkan mask boolean untuk setiap kolom yang menunjukkan nilai-nilai yang termasuk kategori outlier. Setelah itu, jumlah outlier dihitung untuk setiap kolom dengan menjumlahkan mask tersebut. Dataset yang diproses dengan fungsi ini adalah subset kolom numerik dari train_set, yang dipilih menggunakan perintah select_dtypes. Hasil deteksi outlier disimpan dalam variabel outlier_counts, yang menunjukkan jumlah outlier di setiap kolom numerik. Proses ini memberikan

informasi mengenai seberapa besar pengaruh outlier dalam dataset, sehingga memungkinkan pengguna untuk memutuskan langkah selanjutnya dalam menangani outlier, seperti penggantian atau penghapusan.

3. Remove Duplicates

Berikut merupakan potongan code implementasi Remove Duplicates:

```
# Mengecek duplikat hanya untuk nilai yang bukan NaN
duplicates = train_set[train_set['URL'].notna() &
train_set['URL'].duplicated(keep=False)]

# Menampilkan jumlah dan detail duplikat di luar NaN
print("Jumlah Duplikat di Kolom 'URL' (Di Luar NaN):",
duplicates.shape[0])
print("\nDetail Baris Duplikat di Kolom 'URL' (Di Luar NaN):")
print(duplicates)

# Menghapus duplikat hanya untuk nilai yang bukan NaN di kolom
'URL' dan menyimpan baris pertama
train_set=
train_set.loc[train_set['URL'].notna()].drop_duplicates(subset=['
URL'], keep='first')
```

Langkah pertama adalah mendeteksi duplikat dengan menggunakan metode duplicated pada kolom URL yang memiliki nilai valid (bukan NaN). Duplikat ini diidentifikasi dengan parameter keep=False, yang memastikan bahwa semua baris duplikat ditandai tanpa pengecualian. Hasil deteksi disimpan dalam variabel duplicates, dan jumlah serta detail baris duplikat ditampilkan menggunakan perintah print. Selanjutnya, duplikat dihapus dengan memanfaatkan fungsi drop_duplicates, di mana hanya baris pertama dari setiap kelompok duplikat yang dipertahankan menggunakan parameter keep='first'. Proses ini diterapkan hanya pada baris yang memiliki nilai valid di kolom URL, yang difilter dengan kondisi train_set['URL'].notna(). Hasilnya adalah dataset yang bersih dari duplikat, dengan setiap nilai unik di kolom URL dipertahankan. Cara ini bertujuan untuk meningkatkan kualitas data dengan memastikan bahwa hanya satu salinan dari setiap nilai unik di kolom URL yang tersisa.

4. Feature Engineering

Berikut merupakan potongan code implementasi Feature Engineering:

```
columns_to_drop = [
   'FILENAME','id', 'URL', 'Domain', 'TLDLength',
'NoOfObfuscatedChar',
   'NoOfLettersInURL', 'NoOfDegitsInURL', 'NoOfEqualsInURL',
   'NoOfQMarkInURL', 'NoOfAmpersandInURL',
   'LineOfCode', 'LargestLineLength', 'Title',
'URLTitleMatchScore', 'HasFavicon',
   'Robots', 'NoOfSelfRedirect', 'NoOfPopup', 'NoOfiFrame',
'HasExternalFormSubmit',
   'HasSubmitButton', 'HasPasswordField', 'Bank', 'Pay',
'Crypto',
   'NoOfSelfRef', 'NoOfEmptyRef'
]

from sklearn.base import BaseEstimator, TransformerMixin
import pandas as pd

class FeatureDropper(BaseEstimator, TransformerMixin):
    def __init__(self, drop_features=None):
        """
        Parameters:
        - drop_features: list, default=None
          Daftar fitur (kolom) yang akan dihapus dari dataset.
        """
        self.drop_features = drop_features if drop_features else
[]

    def fit(self, X, y=None):
        # Tidak ada fitting yang diperlukan
        return self

    def transform(self, X):
        # Menghapus kolom yang ditentukan
        return X.drop(columns=self.drop_features,
errors='ignore')
```

```
from sklearn.base import BaseEstimator, TransformerMixin
import numpy as np
import pandas as pd

class FeatureCreator(BaseEstimator, TransformerMixin):
    def __init__(self):
        pass
```

```
    def fit(self, X, y=None):
        return self

    def transform(self, X):

        X = X.copy()
        if {'NoOfCSS', 'NoOfJS',
'NoOfImage'}.issubset(X.columns):
            X['TotalPageResource'] = X['NoOfCSS'] + X['NoOfJS'] +
X['NoOfImage']
        else:
            raise ValueError("Kolom 'NoOfCSS', 'NoOfJS', atau
'NoOfImage' tidak ditemukan dalam DataFrame.")
        return X
```

Proses feature engineering dilakukan dengan menghapus kolom-kolom yang dianggap tidak relevan atau tidak memberikan kontribusi signifikan terhadap model machine learning. Pemilihan kolom ini dilakukan berdasarkan analisis domain knowledge atau evaluasi korelasi, di mana kolom-kolom tersebut dianggap tidak relevan atau memiliki korelasi rendah dengan target label.

Untuk mengotomatisasi proses penghapusan fitur, sebuah class bernama FeatureDropper didefinisikan sebagai subclass dari BaseEstimator dan TransformerMixin dari scikit-learn. Class ini dirancang untuk mempermudah penghapusan kolom dengan menyediakan parameter drop_features, yang menerima daftar kolom yang akan dihapus. Pada metode fit, tidak ada tindakan yang dilakukan karena proses ini tidak memerlukan fitting terhadap data. Metode transform adalah inti dari class ini, di mana dataset dimodifikasi dengan menghapus kolom yang telah ditentukan menggunakan fungsi drop dari pandas. Sementara itu, Class FeatureCreator digunakan untuk menambahkan fitur baru bernama TotalPageResource, yang dihitung sebagai jumlah dari kolom NoOfCSS, NoOfJS, dan NoOfImage. Sebagai bagian dari pipeline scikit-learn, class ini memastikan data asli tetap utuh dengan membuat salinan dataset sebelum transformasi. Sebelum menciptakan fitur, class memverifikasi keberadaan kolom yang diperlukan dan memunculkan error jika kolom tidak ditemukan.

Berikut adalah langkah-langkah yang dilakukan pada tahap preprocessing:

1. Feature Scalling

Berikut merupakan potongan code Feature Scalling:

```
from sklearn.preprocessing import StandardScaler
import numpy as np


class FeatureScaler(BaseEstimator, TransformerMixin):
    def __init__(self):
        self.scaler = StandardScaler()
        self.scalable_columns = None


    def fit(self, X, y=None):
            numeric_columns = X.select_dtypes(include=['float64',
'int64'])
                self.scalable_columns = numeric_columns.loc[:,
numeric_columns.nunique() > 2].columns
        self.scaler.fit(X[self.scalable_columns])
        return self


    def transform(self, X):
        X = X.copy()
                                    X[self.scalable_columns] =
self.scaler.transform(X[self.scalable_columns])
        return X
```

Class FeatureScaler untuk melakukan **feature scaling** pada kolom numerik dalam dataset menggunakan StandardScaler dari scikit-learn. Pada metode fit, class pertama-tama mengidentifikasi kolom numerik dalam dataset dengan tipe data float64 atau int64. Kemudian, kolom yang memiliki lebih dari dua nilai unik dipilih sebagai

kolom yang dapat diskalakan, karena kolom dengan dua nilai (biner) tidak memerlukan scaling. StandardScaler kemudian di-fit pada data di kolom-kolom yang telah teridentifikasi. Pada metode transform, dataset disalin untuk memastikan data asli tetap utuh, dan nilai di kolom yang telah di-fit sebelumnya diubah menggunakan scaler untuk menerapkan standardisasi. Standardisasi ini memastikan bahwa setiap kolom memiliki rata-rata 0 dan standar deviasi 1, sehingga menghilangkan dominasi fitur dengan skala besar terhadap fitur lain yang memiliki skala kecil.

2. Feature Encoding

Berikut merupakan potongan code Feature Encoding:

```
from sklearn.preprocessing import  LabelEncoder
class FeatureEncoder(BaseEstimator, TransformerMixin):
    def __init__(self):
        self.label_encoders = {}
        self.onehot_encoder = None
        self.onehot_columns = None

    def fit(self, X, y=None):
                                          categorical_columns     =
X.select_dtypes(include=['object']).columns
        for col in categorical_columns:
            le = LabelEncoder()
            X[col] = le.fit_transform(X[col])
            self.label_encoders[col] = le
        return self

    def transform(self, X):
        X = X.copy()
        for col, le in self.label_encoders.items():
            X[col] = le.transform(X[col])
        return X
```

Class FeatureEncoder untuk melakukan feature encoding pada kolom-kolom kategorikal dalam dataset menggunakan LabelEncoder dari scikit-learn. Pada metode fit, class mengidentifikasi kolom-kolom kategorikal dengan tipe data object. Untuk setiap kolom tersebut, instance LabelEncoder dibuat dan di-fit pada data untuk mengonversi nilai kategorikal menjadi representasi numerik. Encoder untuk setiap kolom disimpan dalam dictionary label_encoders agar dapat digunakan kembali pada proses transformasi. Pada metode transform, dataset disalin untuk menjaga data asli

tetap utuh. Nilai pada kolom-kolom kategorikal yang telah di-fit sebelumnya diubah menjadi representasi numerik menggunakan encoder yang tersimpan. Proses ini memastikan konsistensi dalam encoding data pada dataset baru atau tambahan. Dengan menggunakan FeatureEncoder, data kategorikal dapat dikonversi ke bentuk numerik yang dapat diproses oleh model pembelajaran mesin, sehingga model dapat memahami pola dalam fitur kategorikal dengan lebih efektif.

3. Handling Imbalanced Dataset

Berikut merupakan potongan code Handling Imbalanced Dataset:

```
from sklearn.utils.class_weight import compute_class_weight
from collections import Counter


class ImbalancedHandler(BaseEstimator, TransformerMixin):
    def __init__(self):
        pass

    def fit(self, X, y):
        self.classes = np.unique(y)
            self.class_weights = compute_class_weight('balanced',
classes=self.classes, y=y)
        return self

    def transform(self, X, y):
        class_counts = Counter(y)
        max_class = max(class_counts.values())
        X_resampled = X.copy()
        y_resampled = y.copy()

        for class_label in class_counts:
            if class_counts[class_label] < max_class:
                diff = max_class - class_counts[class_label]
                indices = np.where(y == class_label)[0]
                 sampled_indices = np.random.choice(indices, diff,
replace=True)
                            X_resampled = np.vstack([X_resampled,
X.iloc[sampled_indices]])
                            y_resampled = np.hstack([y_resampled,
y[sampled_indices]])

        return X_resampled, y_resampled
```

Untuk menangani dataset yang tidak seimbang dengan memberikan bobot kelas dan melakukan oversampling pada kelas minoritas pada Class ImbalancedHandler. Pada metode fit, class menghitung bobot untuk setiap kelas menggunakan fungsi compute_class_weight dari scikit-learn. Bobot ini dihitung dengan pendekatan 'balanced', di mana kelas dengan jumlah sampel lebih sedikit diberikan bobot lebih besar untuk mengimbangi distribusi data yang tidak merata. Hasil perhitungan bobot disimpan sebagai atribut class_weights.

Pada metode transform, class menangani ketidakseimbangan dataset dengan oversampling kelas minoritas. Menggunakan library collections.Counter, jumlah sampel untuk setiap kelas dihitung, dan kelas dengan jumlah sampel terbesar diidentifikasi sebagai referensi. Untuk setiap kelas dengan jumlah sampel lebih kecil, perbedaan jumlah sampel dihitung, dan sampel tambahan untuk kelas tersebut diambil secara acak dengan penggantian (replacement). Sampel baru ini ditambahkan ke dataset hingga distribusi antar kelas menjadi seimbang. Dataset hasil oversampling dikembalikan dalam bentuk array untuk fitur (X_resampled) dan target label (y_resampled). Cara ini membuat model machine learning tidak bias terhadap kelas mayoritas.

4. Hasil Prediksi Algoritma

Untuk Algoritma KNN dengan menggunakan library sklearn didapat hasil akurasi 96.45%. Sedangkan untuk algoritma KNN dengan scretch masih mengalami error yang menghasilkan akurasi 0.0%.

Untuk algoritma Naive Bayes menggunakan library sklearn didapat hasil akurasi 50.52%. Sedangkan untuk algoritma Naive bayes dengan scretch belum berhasil untuk diimplementasi.

5. Pembagian Tugas

Nama Mahasiswa	:	Muhammad Daffa Kusuma
NIM	:	18222108

Tugas ke-	Kegiatan
1	Menegerjakan Data PreProcessing
2	
3	
4	
5	

Nama Mahasiswa	:	Ahmad Fawwazi
NIM	:	18222117
Tugas ke-	**Kegiatan**	
1	Mengerjakan Dokumen Laporan	
2	Mengerjakan Error Analysis	
3		
4.		
5.		
6.		

Nama Mahasiswa	:	Muhammad Faishal Putra
NIM	:	18222129
Tugas ke-	**Kegiatan**	
1	Mengerjakan Algoritma SKlearn Naive Bayes dan KNN	
2	Mengerjakan Algoirtuma From Scratch Naive Bayes dan KNN	
3		
4		

Nama Mahasiswa	:	Muhammad Faishal Firdaus
NIM	:	18222136
Tugas ke-	**Kegiatan**	
1	Mengerjakan Split, Pipeline	
2	Mengerjakan Data Cleaning sampai featureCreation	
3		
4		
5		

6. Referensi

https://archive.ics.uci.edu/dataset/967/phiusiil+phishing+url+dataset

https://www.sciencedirect.com/science/article/abs/pii/S0167404823004558?via%3Dihub

https://scikit-learn.org/1.5/modules/neighbors.html

https://scikit-learn.org/1.5/modules/naive_bayes.html