KECERDASAN KOMPUTASIONAL – [IF184503]

FINAL PROJECT

Kelompok 3

Kevin Angga Wijaya	05111840000024
Angelita Titiandes Br. Silalahi	05111840000088
Aflakah Nur Farhana	05111840000120

ABSTRAK

Wine merupakan minuman beralkohol yang dibuat dari fermentasi buah, khususnya anggur. Beberapa faktor dapat menimbulkan kecenderungan seseorang untuk menyukai atau tidak menyukai varian dari wine. Sebuah data-set bernama wine-review merupakan sebuah data set berisi hasil review dari berbagai jenis wine. Di dalam data ini terdapat fitur fitur yang dapat menentukan manakah wine yang paling bagus. Perlu untuk mengetahui manakah fitur di dalam data yang paling mempengaruhi dan memilih algoritma untuk mengoptimalkan tingkat akurasi ketepatannya. Untuk melakukan pengolahan data tersebut digunakan proses data mining. Data mining merupakan suatu metode analisis data. Algoritma data mining yang digunakan adalah, *decision tree*, SVM *(support Vector Machine*), KNN (*K-nearest neighbor*) ANN *(Artificial Neural Network).*

Kata Kunci: Akurasi, ANN, Data mining, *Decision tree*, KNN, wine

DAFTAR ISI

BAB 1.
PENDAHULUAN

1.1. Latar Belakang

Wine adalah minuman beralkohol yang dibuat dari fermentasi buah, khususnya anggur. Anggur dihancurkan, lalu dicampur dengan variasi yeast untuk mengubah kadar gula menjadi alkohol. Dengan variasi spesies anggur, produk wine yang dihasilkan pun akan berbeda dimana hal ini akan mempengaruhi rasa dari wine tersebut. Beberapa faktor menimbulkan kecenderungan seseorang untuk menyukai atau tidak menyukai varian dari wine. Seorang full stack engineering dari Amerika Serikat bernama Zackthoutt membuat sebuah dataset wine-review.

Data wine-review merupakan sebuah data set yang menyimpan data terkait dengan review wine sebanyak 150.000 data. Didalam data ini terdapat 10 fitur yaitu, country, description, designation, points, price, province, region_1, region_2, variety, winery.

Data wine-review dapat digunakan untuk pengambilan keputusan menentukan manakah wine yang paling baik. Di dalam data tersebut belum diketahui manakah sebenarnya fitur yang paling mempengaruhi dari semua fitur yang ada, oleh karenanya harus dilakukan analisis terlebih dahulu terhadap data. Serta perlu untuk dilakukan pencarian metode yang paling tepat guna mendapatkan akurasi yang paling optimal.

Jumlah data yang besar dapat menyebabkan sulitnya melakukan analisis terhadap data, oleh karenanya pada era ini banyak digunakan metode metode data mining yang dapat melakukan pengolahan data secara efisien.
Data mining sendiri sering disebut sebagai *knowledge discovery in database* (KDD) adalah kegiatan yang meliputi pengumpulan pemakaian data historis untuk menemukan keteraturan, pola hubungan dalam set data berukuran besar.
Teknik yang digunakan dalam *data mining* cukup beragam, beberapa diantaranya adalah Decision tree, ANN *(Artificial Neural Network),* SVM, KNN.

Decision tree merupakan metode yang sangat populer untuk digunakan karena hasil dari model yang terbentuk mudah untuk dipahami. Dinamai pohon keputusan karena aturan yang dibentuk mirip dengan bentuk pohon. Pohon dibentuk dari proses penyortiran rekursif biner dalam kelompok data, sehingga nilai variabel respons di setiap kelompok data membuat hasil penyortiran menjadi lebih homogen (Breiman et al. (1984))

Artificial Neural Network (ANN) merupakan sebuah sistem cerdas yang digunakan untuk mengolah informasi yang merupakan perkembangan dari generalisasi model matematika. Prinsip kerja ANN terinspirasi dari prinsip kerja sistem jaringan saraf (neural network) manusia.

Metode Support Vector Machine (SVM) adalah metode klasifikasi linier dengan menemukan hyperplane terbaik yang berfungsi sebagai pemisah dua buah kelas pada inputspace (Saifinnuha, 2015). Prinsip dasar dari SVM adalah pengklasifikasi linear, kemudian dikembangkan menjadi pengklasifikasi nonlinear dengan memasukkan kernel trik pada ruang dimensi tinggi.

Algoritma K-Nearest Neighbor (KNN) adalah sebuah metode untuk melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut (Febri, 2015). Data pembelajaran diproyeksikan ke ruang berdimensi banyak, dimana masing – masing dimensi merepresentasikan fitur dari data.

1.2. Rumusan Masalah

1. Apakah fitur yang paling berpengaruh dan paling tidak berpengaruh terhadap target?
2. Bagaimana cara mendapatkan akurasi paling baik untuk memprediksi data?
3. Bagaimana perbedaan hasil akurasi antara fitur paling berpengaruh dan tidak berpengaruh?

BAB 2.
DESAIN DAN IMPLEMENTASI

2.1. Persiapan Data

Data yang digunakan dalam percobaan ini adalah data Wine-Review. Data ini terdiri dari 150.930. Data ini memiliki 10 fitur yaitu country, description, designation, points, price, province, region_1, region_2, variety, winery. Dalam data ini fitur points akan digunakan sebagai target.

Sebelum Uji Coba dilakukan EDA *(Exploratory Data Analysis)* terhadap data terlebih dahulu.

1. Mengidentifikasi tipe data dari setiap kolom yang ada di dataset.
   Hal ini dilakukan untuk melihat tipe data apa saja yang ada di dataset, agar nantinya bisa diubah sesuai dengan kebutuhan.
2. Identifikasi ukuran dari dataset (jumlah baris dan kolom).
   Hal ini dilakukan untuk memastikan jumlah baris dan kolom yang tersisa ketika sudah dilakukan pembersihan sebuah kebutuhan, dan juga untuk membandingkan jumlah data saat sebelum dan sesudah dibersihkan.
3. Melihat ringkasan data numerik.
   Hal ini dilakukan untuk melihat deskripsi dari data numerik, termasuk jumlah datanya. Dengan melihat jumlah datanya, kita bisa mengidentifikasi keberadaan null value dengan membandingkan jumlah data di deskripsi dengan jumlah data keseluruhan.
4. Menghitung jumlah instances/data dari setiap value.
   Hal ini dilakukan untuk melihat seberapa banyak data untuk setiap jenis value yang ada. Dengan ini kita bisa menemukan variabel yang memiliki terlalu banyak data yang unik yang nantinya bisa mengarahkan kita kepada keputusan untuk menggunakan atau tidak menggunakan masing-masing kolom dari dataset.
5. Mengidentifikasi value dari setiap kolom.
   Hal ini dilakukan untuk melihat jumlah value yang unik dari setiap kolom, dan menunjukkan apa saja value tersebut. Dengan ini kita bisa melihat kolom mana yang memiliki value unik yang banyak atau sedikit

yang nantinya bisa mengarahkan kita kepada keputusan untuk menggunakan atau tidak menggunakan masing-masing kolom dari dataset.

6. Mengidentifikasi jumlah null value pada setiap kolom.
   Hal ini dilakukan untuk melihat berapa banyak null value dari setiap kolom data, agar bisa menjadi pertimbangan untuk menggunakan atau tidak masing-masing kolom dari dataset.
7. Melihat persebaran data dengan menggunakan histogram
   Hal ini dilakukan untuk melihat persebaran data dan mengidentifikasi outlier yang ada pada data.

Dari hasil yang didapatkan, maka diputuskan beberapa hal sebagai berikut:

1. Menghapus kolom description, karena kolom description memiliki sangat banyak value yang unik, yakni 97.821 data. Jadi hampir sebagian besar data tidak memiliki duplikat.
2. Menghapus kolom region_2, karena sebanyak 89.977 datanya kosong. Sehingga jika dilakukan penghapusan null value akan mengurangi sangat banyak data yang ada.
3. Designation, region_1, dan juga price juga memiliki banyak null value. Tetapi ketika datanya dibersihkan, masih cukup banyak data yang tersisa, sehingga kolom tersebut dibiarkan. Selain itu, ketiga kolom ini juga berdampak cukup besar terhadap kolom nilai.

2.2. Skenario Uji Coba

Langkah awal

1. Membersihkan data

```
df.drop(columns=['description', 'Unnamed: 0', 'region_2'], inplace=True)
df
```

2. Remove NULL Value

Instances/row yang memiliki null value di salah satu kolomnya dibuang

```
for col in df.columns:
    df = df[df[col].notnull()]
df
```

3. Mengambil data yang pricenya di bawah 1000, dikarenakan data yang di atas 1000 hanya ada 4 data dan menjadi outlier.

```
harga_kurang_1000 = df.price < 1000
df = df[harga_kurang_1000]
df
```

4. Mendiskritkan target menjadi kontinu menggunakan qcut dari pandas dengan parameter pd.qcut(kolom, jumlah_kelas, label) dan replace kolom target (point) dengan angka

```
df_c = df.copy()
df_c['points'] = pd.qcut(df_c['points'], 2, labels=['low', 'high'])
df_c['points'].replace({'low': 0, 'high': 1}, inplace=True)
df_c
```

2.2.1 Skenario 1

Berikut ini merupakan langkah langkah untuk menemukan fitur yang paling berpengaruh dan paling tidak berpengaruh dari data.

1. Deklarasi fungsi menghitung entropy

```
def calc_entropy(column):
    """
    Calculate entropy given a pandas series, list, or numpy array.
    """
    # Compute the counts of each unique value in the column
    counts = np.bincount(column)
    # Divide by the total column length to get a probability
    probabilities = counts / len(column)

    # Initialize the entropy to 0
    entropy = 0
    # Loop through the probabilities, and add each one to the total entropy
    for prob in probabilities:
        if prob > 0:
            # use log from math and set base to 2
            entropy += prob * math.log(prob, 2)

    return -entropy
```

2. Deklarasi fungsi menghitung information gain

```
def calc_information_gain(data, split_name, target_name):
    """
    Calculate information gain given a data set, column to split on, and target
    """
    # Calculate the original entropy
    original_entropy = calc_entropy(data[target_name])

    #Find the unique values in the column
    values = data[split_name].unique()


    # Make two subsets of the data, based on the unique values
    left_split = data[data[split_name] == values[0]]
    right_split = data[data[split_name] == values[1]]

    # Loop through the splits and calculate the subset entropies
    to_subtract = 0
    for subset in [left_split, right_split]:
        prob = (subset.shape[0] / data.shape[0])
        to_subtract += prob * calc_entropy(subset[target_name])

    # Return information gain
    return original_entropy - to_subtract
```

3. Define fungsi untuk me-return hasil information gain seluruh kolom

```
columns = ['country', 'designation', 'price', 'province', 'region_1', 'variety', 'winery']

def print_info_gain(columns):
  #Intialize an empty dictionary for information gains
  information_gains = {}

  #Iterate through each column name in our list
  for col in columns:
    #Find the information gain for the column
    information_gain = calc_information_gain(df_c, col, 'points')
    #Add the information gain to our dictionary using the column name as the ekey
    information_gains[col] = information_gain
  key=information_gains.get
  #Return the key with the highest value
  return (information_gains, key)
```

4. Define fungsi untuk me-return hasil information gain tertinggi

```
def highest_info_gain(columns):
  #Intialize an empty dictionary for information gains
  information_gains = {}

  #Iterate through each column name in our list
  for col in columns:
    #Find the information gain for the column
    information_gain = calc_information_gain(df_c, col, 'points')
    #Add the information gain to our dictionary using the column name as the ekey
    information_gains[col] = information_gain
  #Return the key with the highest value
  return max(information_gains, key=information_gains.get)
```

5. Define fungsi untuk me-return hasil information gain terendah

```
def lowest_info_gain(columns):
  #Intialize an empty dictionary for information gains
  information_gains = {}

  #Iterate through each column name in our list
  for col in columns:
    #Find the information gain for the column
    information_gain = calc_information_gain(df_c, col, 'points')
    #Add the information gain to our dictionary using the column name as the ekey
    information_gains[col] = information_gain

  #Return the key with the highest value
  return min(information_gains, key=information_gains.get)
```

6. Menggunakan Target Encoder yang meng-encode berdasarkan target. Value yang diberikan antara 0 sampai 1

```
te = TargetEncoder()
```

```
y = df_c['points'].values
X = df_c.drop(columns=['points']).values
# X = df_disc[['Make', 'Engine HP', 'Engine Cylinders', 'Year', 'highway MPG', 'city mpg', 'Engine Fuel Type']].values
te.fit(X, df_c['points'].values)
X = te.transform(X)

X.columns = ['country', 'designation', 'price', 'province', 'region_1', 'variety', 'winery']

X
```

7. Mencetak Information gain seluruh kolom

```
info_gain = print_info_gain(columns)
for i in info_gain[0]:
  print(i ,":",info_gain[0][i])
```

8. Mencetak Information gain tertinggi

```
print(highest_info_gain(columns))
```

9. Mencetak Information gain terendah

```
print(lowest_info_gain(columns))
```

2.2.2 Skenario 2

Berikut merupakan langkah langkah untuk menentukan manakah metode paling optimal yang dapat digunakan. Metode yang digunakan antara lain ANN,Decision tree, SVM, KNN.

1. Melakukan encoding

   Menggunakan Target Encoder yang meng-encode berdasarkan target. Value yang diberikan antara 0 sampai 1

```
te = TargetEncoder()

y = df_disc['points'].values
X = df_disc.drop(columns=['points']).values
te.fit(X, y)
X = te.transform(X)

X.head()
```

2. Decision tree

   Pembuatan parameter yang akan di uji

```
param_dt = {
    'test_size': [0.9, 0.7, 0.5, 0.3, 0.1],
    'max_depth': [3, 5, 7],
    'criterion': ['gini', 'entropy']
}
```

Menjalankan Decision Tree sekaligus didapatkan variasi untuk hasil akurasi yang paling besar.

```
row_format ="{:>10}" * (len(param_dt) + 2)
print(row_format.format('t_size', 'max_depth', 'criterion', 'akurasi', 'f1'))
acc_bfr = -math.inf
for i in param_dt['test_size']:
  for j in param_dt['max_depth']:
    for k in param_dt['criterion']:
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=i, random_state=42)
      dt = DecisionTreeClassifier(random_state=42, max_depth=j, criterion=k)
      dt.fit(X_train, y_train)

      # print(y_test)
      pred = dt.predict(X_test)
      acc = metrics.accuracy_score(y_test, pred)
      f1 = metrics.f1_score(y_test, pred)
      if(acc > acc_bfr):
        acc_bfr = acc
        best_param_dt = {'t_size': i, 'max_depth': j, 'criterion': k, 'accuracy': acc, 'f1': f1}
        best_dt = dt
      print(row_format.format(str(i), str(j), str(k), '%.3f' % acc, '%.3f' % f1))
print('best param: ', best_param_dt)
```

3. SVM

Pembuatan parameter yang akan di uji

```
param_svm = {
  'test_size': [0.9, 0.7, 0.5, 0.3, 0.1],
  'C'    : [1, 10, 100, 0.1],
  'gamma': [0.1, 0.01, 0.001, 1.0],
  'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
}
```

Menjalankan SVM sekaligus didapatkan variasi untuk hasil akurasi yang paling besar

```
row_format ="{:>10}" * (len(param_svm) + 2)
print(row_format.format(*param_svm, 'akurasi', 'f1'))
acc_bfr = -math.inf
for i in param_svm['test_size']:
  for j in param_svm['C']:
    for k in param_svm['gamma']:
      for l in param_svm['kernel']:
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=i, random_state=42)
        svc = SVC(C=j, gamma=k, kernel=l)
        svc.fit(X_train, y_train)

        # print(y_test)
        pred = svc.predict(X_test)
        acc = metrics.accuracy_score(y_test, pred)
        f1 = metrics.f1_score(y_test, pred)
        if(acc > acc_bfr):
          acc_bfr = acc
          best_param_svm = {'t_size': i, 'C': j, 'gamma': k, 'kernel': l, 'accuracy': acc, 'f1': f1}
        print(row_format.format(str(i), str(j), str(k), str(l), '%.3f' % acc, '%.3f' % f1))
print('best param: ', best_param_svm)
```

4. KNN

Pembuatan parameter yang akan di uji

```
param_knn = {
  'test_size': [0.9, 0.7, 0.5, 0.3, 0.1],
  'n_neighbor': [3, 5],
  'weights': ['uniform', 'distance'],
  'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
}
```

Menjalankan KNN sekaligus didapatkan variasi untuk hasil akurasi yang paling besar

```
row_format ="{:>10}" * (len(param_knn) + 2)
print(row_format.format(*param_knn, 'akurasi', 'f1'))
acc_bfr = -math.inf
for i in param_knn['test_size']:
  for j in param_knn['n_neighbor']:
    for k in param_knn['weights']:
      for l in param_knn['algorithm']:
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=i, random_state=42)
        knn = KNeighborsClassifier(n_neighbors=j, weights=k, algorithm=l)
        knn.fit(X_train, y_train)

        # print(y_test)
        pred = knn.predict(X_test)
        acc = metrics.accuracy_score(y_test, pred)
        f1 = metrics.f1_score(y_test, pred)
        if(acc > acc_bfr):
          acc_bfr = acc
          best_param_knn = {'t_size': i, 'n_beighbors': j, 'weights': k, 'algorithm': l, 'accuracy': acc, 'f1': f1}
        print(row_format.format(str(i), str(j), str(k), str(l), '%.3f' % acc, '%.3f' % f1))
print('best param: ', best_param_knn)
```

5. ANN

Berikut ini merupakan langkah langkah dalam melakukan analisis terhadap parameter yang terdapat pada ANN sehingga bisa memperoleh hasil akurasi paling optimal.

1. Membuat skenario model uji coba

Model	Hidden-1		Hidden-2		Hidden-3		Learning Rate	Epochs	Optimizers	%train
	Activation	Nodes	Activation	Nodes	Activation	Nodes				
Model 1	selu	512	selu	256	selu	128	0,005	500	adam	50%
Model 2	tanh	512	tanh	256	tanh	128	0,005	500	adam	50%
Model 3	relu	512	relu	256	relu	128	0,005	500	adam	50%
Model 4	relu	512	relu	256	relu	128	0,005	500	Adadelta	50%
Model 5	relu	512	relu	256	relu	128	0,005	500	adam	30%
Model 6	relu	512	relu	256	relu	128	0,005	100	adam	30%
Model 7	relu	512	relu	256	-	-	0,005	500	adam	30%
Model 8	relu	256	relu	128	-	-	0,005	500	adam	30%
Model 9	relu	256	-	-	-	-	0,005	500	adam	30%
Model 10	relu	512	-	-	-	-	0,005	500	adam	30%
Model 11	relu	512	relu	256	relu	128	0,1	500	adam	30%

2. Melakukan definisi test size

   Dilakukan variasi dari jumlah data tes dan data train yang digunakan guna menemukan jumlah yang dapat mengoptimalkan akurasi.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, random_state=1)
```

3. Melakukan definisi activation, nodes, hidden layer

   Dilakukan variasi dari jumlah node dan hidden layer guna menemukan jumlah yang dapat mengoptimalkan akurasi. Variasi juga dilakukan untuk activation. Activation function berfungsi untuk menentukan apakah neuron tersebut harus "aktif" atau tidak berdasarkan dari weighted sum dari input. Activation yang digunakan adalah relu, selu, tanh. Ketiga aktivasi ini dipilih berdasarkan penelitian terdahulu yang telah dilakukan, bahwa ketiga activation tersebut merupakan yang activation yang paling disarankan.

```
model=keras.models.Sequential([
    keras.layers.Dense(512, input_dim = X_train.shape[1], activation='relu'),
    keras.layers.Dense(units=256, activation='relu'),

    keras.layers.Dense(units=1, activation="sigmoid"),
],name="Initial_model",)
model.summary()
```

4. Melakukan definisi learning rate dan optimizer yang digunakan

   Dilakukan variasi dari jumlah learning rate dan optimizer guna menemukan jumlah yang dapat mengoptimalkan akurasi. Optimizer yang digunakan adalah Adam,Adadelta. Berdasarkan penelitian terdahulu disebutkan bahwa asam merupakan optimizer yang paling optimal dan adadelta merupakan optimizer yang paling cepat.

```
learning_rate = 0.005
optimizer = keras.optimizers.Adam(lr=learning_rate)
```

5. Definisi metrics akurasi yang digunakan dan *loss*

```
model.compile(optimizer=optimizer,
              loss='binary_crossentropy', metrics=['accuracy'])
```

6. Fungsi early stopping

   Berfungsi untuk menghentikan training ketika kerugian mulai meningkat atau dengan kata lain keakuratan validasi mulai menurun.

```
early_stopping_monitor = EarlyStopping(
    monitor='val_loss',
    min_delta=0,
    patience=1000,
    verbose=0,
    mode='auto',
    baseline=None,
    restore_best_weights=True
)
```

7. Menjalankan ANN

   Dilakukan variasi dari epochs guna menemukan jumlah yang dapat mengoptimalkan akurasi

```
history = model.fit(X_train, y_train,
                    epochs=500, batch_size=1024,
                    validation_data=(X_test, y_test),
                    verbose=1, callbacks=[early_stopping_monitor])
```

2.2.3 Skenario 3

Berikut merupakan langkah langkah untuk melihat perbedaan hasil akurasi yang didapatkan ketika menggunakan fitur fitur yang paling berpengaruh dengan fitur yang tidak berpengaruh. Fitur berpengaruh yang digunakan adalah ‘designation’,’winery’, ‘price’. Fitur tidak berpengaruh yang digunakan adalah ‘country’ dan ‘province’. Metode yang digunakan antara lain ANN, Decision tree, SVM, KNN.

1. Melakukan drop kolom

   Lakukan drop kolom, sisakan fitur fitur yang diperlukan saja (variasi versi fitur berpengaruh, variasi versi fitur tidak berpengaruh).

2. ANN

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.7, random_state=1)
```

```
model=keras.models.Sequential([
    keras.layers.Dense(512, input_dim = X_train.shape[1], activation='relu'),
    keras.layers.Dense(units=256, activation='relu'),
    keras.layers.Dense(units=128, activation='relu'),

    keras.layers.Dense(units=1, activation="sigmoid"),
],name="Initial_model",)
model.summary()
```

```
learning_rate = 0.005
optimizer = keras.optimizers.Adam(lr=learning_rate)
```

```
model.compile(optimizer=optimizer,
            loss='binary_crossentropy', metrics=['accuracy'])
```

```
early_stopping_monitor = EarlyStopping(
    monitor='val_loss',
    min_delta=0,
    patience=1000,
    verbose=0,
    mode='auto',
    baseline=None,
    restore_best_weights=True
)
```

```
history = model.fit(X_train, y_train,
                    epochs=500, batch_size=1024,
                    validation_data=(X_test, y_test),
                    verbose=1, callbacks=[early_stopping_monitor])
```

3. Decision tree

   Pembuatan parameter yang akan di uji

```
param_dt = {
    'test_size': [0.9, 0.7, 0.5, 0.3, 0.1],
    'max_depth': [3, 5, 7],
    'criterion': ['gini', 'entropy']
}
```

Menjalankan Decision Tree sekaligus didapatkan variasi untuk hasil akurasi yang paling baik

```
row_format ="{:>10}" * (len(param_dt) + 2)
print(row_format.format('t_size', 'max_depth', 'criterion', 'akurasi', 'f1'))
acc_bfr = -math.inf
for i in param_dt['test_size']:
  for j in param_dt['max_depth']:
    for k in param_dt['criterion']:
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=i, random_state=42)
      dt = DecisionTreeClassifier(random_state=42, max_depth=j, criterion=k)
      dt.fit(X_train, y_train)

      # print(y_test)
      pred = dt.predict(X_test)
      acc = metrics.accuracy_score(y_test, pred)
      f1 = metrics.f1_score(y_test, pred)
      if(acc > acc_bfr):
        acc_bfr = acc
        best_param_dt = {'t_size': i, 'max_depth': j, 'criterion': k, 'accuracy': acc, 'f1': f1}
        best_dt = dt
      print(row_format.format(str(i), str(j), str(k), '%.3f' % acc, '%.3f' % f1))
print('best param: ', best_param_dt)
```

4. SVM

Pembuatan parameter yang akan di uji

```
param_svm = {
  'test_size': [0.9, 0.7, 0.5, 0.3, 0.1],
  'C'    : [1, 10, 100, 0.1],
  'gamma': [0.1, 0.01, 0.001, 1.0],
  'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
}
```

Menjalankan SVM sekaligus didapatkan variasi untuk hasil akurasi yang paling baik

```
row_format ="{:>10}" * (len(param_svm) + 2)
print(row_format.format(*param_svm, 'akurasi', 'f1'))
acc_bfr = -math.inf
for i in param_svm['test_size']:
  for j in param_svm['C']:
    for k in param_svm['gamma']:
      for l in param_svm['kernel']:
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=i, random_state=42)
        svc = SVC(C=j, gamma=k, kernel=l)
        svc.fit(X_train, y_train)

        # print(y_test)
        pred = svc.predict(X_test)
        acc = metrics.accuracy_score(y_test, pred)
        f1 = metrics.f1_score(y_test, pred)
        if(acc > acc_bfr):
          acc_bfr = acc
          best_param_svm = {'t_size': i, 'C': j, 'gamma': k, 'kernel': l, 'accuracy': acc, 'f1': f1}
        print(row_format.format(str(i), str(j), str(k), str(l), '%.3f' % acc, '%.3f' % f1))
print('best param: ', best_param_svm)
```

5. KNN

Pembuatan parameter yang akan di uji

```
param_knn = {
  'test_size': [0.9, 0.7, 0.5, 0.3, 0.1],
  'n_neighbor': [3, 5],
  'weights': ['uniform', 'distance'],
  'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
}
```

Menjalankan KNN sekaligus didapatkan variasi untuk hasil akurasi yang paling baik

```
row_format ="{:>10}" * (len(param_knn) + 2)
print(row_format.format(*param_knn, 'akurasi', 'f1'))
acc_bfr = -math.inf
for i in param_knn['test_size']:
  for j in param_knn['n_neighbor']:
    for k in param_knn['weights']:
      for l in param_knn['algorithm']:
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=i, random_state=42)
        knn = KNeighborsClassifier(n_neighbors=j, weights=k, algorithm=l)
        knn.fit(X_train, y_train)

        # print(y_test)
        pred = knn.predict(X_test)
        acc = metrics.accuracy_score(y_test, pred)
        f1 = metrics.f1_score(y_test, pred)
        if(acc > acc_bfr):
          acc_bfr = acc
          best_param_knn = {'t_size': i, 'n_beighbors': j, 'weights': k, 'algorithm': l, 'accuracy': acc, 'f1': f1}
        print(row_format.format(str(i), str(j), str(k), str(l), '%.3f' % acc, '%.3f' % f1))
print('best param: ', best_param_knn)
```

BAB 3.

HASIL UJI COBA DAN DISKUSI

3.1. Hasil Uji Coba Skenario 1

```
print(highest_info_gain(columns))
```

```
designation
```

```
print(lowest_info_gain(columns))
```

```
country
```

```
print(print_info_gain(columns))
```

Hasil gain dalam bentuk tabel dari gain terendah ke tertinggi:

Fitur	Gain
country	0.404
province	0.579
variety	0.910
region_1	0.944
price	0.991
winery	0.992
designation	0.993

3.2. Hasil Uji Coba Skenario 2

1. Decision Tree

```
best param:  {'t_size': 0.7, 'max_depth': 7, 'criterion':
'entropy', 'accuracy': 0.8776609442060086, 'f1':
0.866683815447934}

time: 2.54 s
```

2. SVM

```
best param:  {'t_size': 0.7, 'C': 10, 'gamma': 1.0,
'kernel': 'rbf', 'accuracy': 0.8807296137339056, 'f1':
0.8662141344117081}

time: 2h 57min 46s
```

3. KNN

```
best param:  {'t_size': 0.3, 'n_beighbors': 5, 'weights':
'distance', 'algorithm': 'auto', 'accuracy':
0.8946024434207891, 'f1': 0.8819074333800841}

time: 6min 8s
```

4. ANN

Model	ACC Dtrain	ACC Dtest	Waktu train
Model 1	0,883	0,863	25min 48s
Model 2	0,876	0,864	24min 41s
Model 3	0,961	0,859	24min 11s
Model 4	0,868	0,862	24min 10s
Model 5	0,963	0,847	18min 48s
Model 6	0,881	0,867	3min 39s
Model 7	0,969	0,852	17min 27s
Model 8	0,955	0,846	8min 54s
Model 9	0,920	0,853	7min 24s

Model 10	0,937	0,839	12min 17s
Model 11	0,552	0,550	18min 50s

Berikut merupakan tabel hasil akurasi data keseluruhan :

Metode	Akurasi
ANN	0,969
Decision tree	0,877
SVM	0,880
KNN	0,894

3.3.Hasil Uji Coba Skenario 3

A. Menggunakan fitur paling berpengaruh

1. ANN

```
Train: 0.901, Test: 0.877

time: 9 min 38 s
```

2. Decision tree

```
best param:  {'t_size': 0.1, 'max_depth': 7, 'criterion':
'gini', 'accuracy': 0.8852118695811454, 'f1':
0.8724905046120457}

time: 2.27 s
```

Parameter terbaik dari decision tree adalah ketika menggunakan data train sebesar 90% dari data, dengan kedalaman 7, criterion yang digunakan adalah GINI menghasilkan akurasi 0.885 dengan runtime 2.52s

3. SVM

```
best param:  {'t_size': 0.1, 'C': 10, 'gamma': 1.0,
'kernel': 'rbf', 'accuracy': 0.8832580290633777, 'f1':
0.866778149386845}

time: 3h 22min 14s
```

Parameter terbaik dari algoritma SVM adalah ketika menggunakan data train sebesar 90% dengan gamma 1.0, kernel yang digunakan adalah RBF menghasilkan akurasi sebesar 0.883 dengan runtime selama 3 jam 22 menit 14 detik.

4. KNN

```
best param:  {'t_size': 0.1, 'n_beighbors': 5, 'weights':
'distance', 'algorithm': 'auto', 'accuracy':
0.9013310538527293, 'f1': 0.8864530635188307}

time: 6min 57s
```

Parameter terbaik saat penggunaan algoritma KNN adalah saat data train yang digunakan sebesar 90% dengan K = 5, weight yang digunakan adalah distance dengan algoritma auto, menghasilkan akurasi sebesar 0.901 dengan runtime selama 6 menit 8 detik.

Berikut merupakan tabel hasil data diurutkan dari algoritma dengan akurasi terbesar ke terkecil:

Metode	Akurasi
ANN	0,901
KNN	0,901
Decision tree	0,885

SVM	0,883

B. Menggunakan fitur paling tidak berpengaruh

1. ANN

```
Train: 0.640, Test: 0.642

time: 17 menit 2 s
```

2. Decision tree

```
best param:  {'t_size': 0.1, 'max_depth': 5,
'criterion': 'gini', 'accuracy': 0.6445755185897011,
'f1': 0.4918032786885245}

time: 2.56 s
```

Parameter terbaik dari decision tree adalah ketika menggunakan data train sebesar 90% dari data, dengan kedalaman 5, criterion yang digunakan adalah GINI menghasilkan akurasi 0.644 dengan runtime 2.56s

3. SVM

Tidak didapatkan hasil. Waktu runtime telah dijalankan selama lebih dari 12 jam hingga menyebabkan session habis pada google collab, sehingga dapat diprediksi bahwa algoritma svm merupakan algoritma yang tidak efektif untuk digunakan dalam data ini.

4. KNN

```
best param:  {'t_size': 0.5, 'n_beighbors': 5,
'weights': 'uniform', 'algorithm': 'brute', 'accuracy':
0.6278283691891677, 'f1': 0.48381225525343335}

time: 44min 36s
```

Parameter terbaik saat penggunaan algoritma KNN adalah saat data train yang digunakan sebesar 50% dengan K = 5, weight yang digunakan adalah uniform dengan algoritma brute menghasilkan akurasi sebesar 0.627 dengan runtime selama 44 menit 36 detik.

Berikut merupakan tabel hasil data diurutkan dari algoritma dengan akurasi terbesar ke terkecil:

Metode	Akurasi
ANN	0,640
Decision tree	0,644
KNN	0,627
SVM	-

3.4.Hasil Uji Coba Skenario 4

Pembahasan dan Kesimpulan

1. Pada decision tree digunakan information gain sebagai acuan dalam pembuatan pohon. Fitur dengan information gain tertinggi akan menjadi root dari *tree*, oleh karenanya fitur dengan gain tertinggi merupakan fitur yang paling berpengaruh karena akan berperan sebagai sebuah root. Maka berlaku sebaliknya untuk fitur yang paling tidak berpengaruh.

   Dari hasil percobaan tersebut dapat ditarik kesimpulan bahwa Fitur yang paling berpengaruh adalah ‘Designation’ karena memiliki gain paling besar, yaitu 0.992. Fitur yang paling tidak berpengaruh adalah ‘Country’ karena memiliki gain paling kecil, yaitu 0.404.

2. Dalam percobaan kali ini, yang dimaksud dengan hasil yang optimal adalah variasi parameter yang dapat menghasilkan hasil dengan akurasi tinggi, waktu runtime yang cenderung sebentar, banyak nya data train yang digunakan. Berdasarkan pertimbangan tersebut diperoleh untuk data wine-review variasi parameter terbaik adalah

1. Decision Tree

   Test size : 70%
   Max depth : 7
   Criterion : Entropy
   Akurasi : 0,877

2. SVM

   Test size : 70%
   C : 10
   Gamma : 1.0
   Kernel : RBF
   Akurasi : 0,880

3. KNN

   Test size : 30%
   K : 5
   Weights : Distance
   Algoritma : Auto
   Akurasi : 0,894

4. ANN

Model 7	0,969	0,852	17min 27s

- Hidden layer : 2

  Semakin banyak hidden layer yang digunakan, maka waktu yang dibutuhkan semakin lama, dan akurasi yang dihasilkan semakin bertambah.

- Node1 : 512

  Semakin besar nilai node akan semakin mempengaruhi akurasi dan waktu, dimana keduanya akan sama sama meningkat.

- Node2 : 256

  Semakin besar nilai node akan semakin mempengaruhi akurasi dan waktu, dimana keduanya akan sama sama meningkat.

- Epochs :500

  Semakin besar epochs dapat meningkatkan tingkat akurasi, namun waktu yang dibutuhkan juga semakin meningkat.

- Learning rate : 0.005

  Tingkat akurasi akan menjadi semakin besar apabila nilai dari learning rate semakin kecil

- Data train : 30%, Data test : 70%

  Semakin besar data train yang dilakukan dapat meningkatkan tingkat akurasi, namun hal tersebut juga menyebabkan meningkatnya waktu yang dibutuhkan.

- waktu : 17 menit 27 detik
- Activation : Relu

  Activation relu merupakan jenis activation yang paling banyak direkomendasikan untuk digunakan dalam rangka pencapaian hasil yang optimal. ReLU dapat mempercepat proses konvergensi yang dilakukan dengan stochastic gradient descent jika dibandingkan dengan sigmoid / tanh. Jika dibandingkan dengan sigmoid/tanh yang memiliki operasi-operasi yang “expensive” (exponentials, etc.), ReLU bisa diimplementasikan hanya dengan membuat pembatas(threshold) pada bilangan nol.

- Optimizer : Adam

  Adam merupakan algoritma yang populer di bidang data mining karena memperoleh hasil yang baik dengan cepat. Algoritma ini melakukan pengoptimalan yang dapat digunakan sebagai pengganti

prosedur penurunan stochastic gradient descent untuk memperbarui bobot jaringan secara berulang-ulang berdasarkan data pelatihan.

Selain menyimpan rata-rata secara eksponensial dari gradien kuadrat sebelumnya, seperti Adadelta, Adam juga mempertahankan rata-rata eksponensial gradien lama.

Adadelta dianggap sebagai optimizer yang cepat namun tidak mempertahankan rata rata eksponensial gradient lama seperti adam, oleh karena itu adam jauh lebih banyak disarankan untuk digunakan daripada adadelta.

Jadi, metode yang menghasilkan akurasi tinggi adalah ANN atau *artificial neural network* sebesar 0,969. Hal ini disebabkan karena pada percobaan ini digunakan banyak data. Semakin banyak data yang digunakan untuk percobaan, maka akan semakin baik pula akurasi prediksi targetnya.

Algoritma ANN lebih unggul dibandingkan dengan algoritma lainnya karena dalam ANN iterasi yang dilakukan jauh lebih banyak dan dilakukan pengecekan dalam komputasinya, sedangkan algoritma lainnya hanya melakukan perhitungan untuk mendapatkan hasil. ANN juga melakukan komputasi yang lebih cepat dibanding algoritma yang lain

3. Akurasi dengan menggunakan data dengan fitur fitur berpengaruh cenderung tinggi dan waktu runtime yang dilakukan cepat. Sebaliknya akurasi dengan fitur fitur tidak berpengaruh memiliki akurasi rendah dan waktu runtime yang lama.

DAFTAR PUSTAKA

B. Santosa, Data Mining. Teknik Pemanfaatan Data untuk Keperluan Bisnis, First Edition ed. Yogyakarta: Graha Ilmu, (2007)

K. Arai and A. R. Barakbah, "Hierarchical K-means: an algorithm for centroids initialization for Kmeans," (2007)

Breiman, L., Friedman, JH., Olshen, RA., Stone, CJ.,"*Classification and Regression Trees"*,(1984)

Saifinnuha, A.Z., 2015. Penerapan Sentimen Analisis pada Twitter Berbahasa Indonesia untuk Mendapatkan Rating Program Televisi Menggunakan Metode Support Vector Machine. Malang: Universitas Brawijaya.

Liantoni, Febri. 2015. Klasifikasi Daun Dengan Perbaikan Fitur Citra Menggunakan Metode K-Nearest Neighbor. Surabaya: Ultimatics.

Ryandhi, Rizky. 2017. PENERAPAN METODE ARTIFICIAL NEURAL NETWORK (ANN) UNTUK PERAMALAN INFLASI DI INDONESIA. Surabaya: ITS.

https://medium.com/@samuelsena/pengenalan-deep-learning-8fbb7d8028ac

https://www.kdnuggets.com/2020/12/optimization-algorithms-neural-networks.html

https://ieeexplore.ieee.org/document/8407425

https://lms.onnocenter.or.id/wiki/index.php/Keras:_Introduction_to_the_Adam_Optimization_Algorithm

https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

https://medium.com/@opam22/menilik-activation-functions-7710177a54c9