DAFTAR PUSTAKA

Aggarwal, C. C. (2022). Machine Learning for Text. *Machine Learning for Text*. https://doi.org/10.1007/978-3-030-96623-2

Campbell, C., & Yiming, Y. (2011). *Learning with Support Vector Machines (Synthesis Lectures on Artificial Intelligence and Machine Learning)*.

Cao, Q., La, L., Liu, H., & Han, S. (2018). Mixed weighted KNN for imbalanced datasets. *International Journal of Performability Engineering*, *14*(7), 1391–1400. https://doi.org/10.23940/ijpe.18.07.p2.13911400

Cunningham, P., & Delany, S. J. (2021). K-Nearest Neighbour Classifiers-A Tutorial. *ACM Computing Surveys*, *54*(6). https://doi.org/10.1145/3459665

Dam, S. M., & Dam, T. C. (2021). *Relationships between Service Quality , Brand Image , Customer Satisfaction , and Customer Loyalty*. *8*(3), 585–593. https://doi.org/10.13106/jafeb.2021.vol8.no3.0585

DjajaPutra, I. O., Prilianti, K. R., & Tirma Irawan, P. L. (2020). Implementasi Text Mining Untuk Analisis Opini Masyarakat Terhadap Kinerja Layanan Transportasi Online Dengan Analisis Faktor. *Jurnal Simantec*, *8*(2), 45–53. https://doi.org/10.21107/simantec.v8i2.6764

Géron, A. (2019). Hands-on Machine Learning whith Scikit-Learing, Keras and Tensorfow. In *O'Reilly Media, Inc*.

Goh, R. Y., & Lee, L. S. (2019). Credit Scoring: A Review on Support Vector Machines and Metaheuristic Approaches. *Advances in Operations Research*, *2019*. https://doi.org/10.1155/2019/1974794

Muttaqin, M. N., & Kharisudin, I. (2021). Analisis Sentimen Pada Ulasan Aplikasi

Gojek Menggunakan Metode Support Vector Machine dan K Nearest Neighbor. *UNNES Journal of Mathematics*, *10*(2), 22–27. http://journal.unnes.ac.id/sju/index.php/ujm

Pisner, D. A., & Schnyer, D. M. (2019). Support vector machine. *Machine Learning: Methods and Applications to Brain Disorders*, 101–121. https://doi.org/10.1016/B978-0-12-815739-8.00006-7

Prabowo, W. A., & Wiguna, C. (2021). Sistem Informasi UMKM Bengkel Berbasis Web Menggunakan Metode SCRUM. *Jurnal Media Informatika Budidarma*, *5*(1), 149. https://doi.org/10.30865/mib.v5i1.2604

Rohwinasakti, S., Irawan, B., & Setianingsih, C. (2021). Sentiment Analysis on Online Transportation Service Products Using K-Nearest Neighbor Method. *Proceedings of the International Conference on Computer, Information, and Telecommunication Systems, CITS 2021*, *7*(3), 9312–9321. https://doi.org/10.1109/CITS52676.2021.9618301

Schoot Uiterkamp, L. (2019). *Improving text representations for NLP from bags to strings of words*. *august*. http://essay.utwente.nl/79245/

Van Der Merwe, A. (1972). Management marketing. In *Agrekon* (Vol. 11, Issue 1). https://doi.org/10.1080/03031853.1972.9523871

LAMPIRAN

Lampiran 1. *Preprocessing* Data

```
def preprocess(text):
  text = text.lower()
  text = re.sub(r'[^a-zA-Z]',' ',str(text))
  text = re.sub(r'\s\s+', ' ', text)
  text = text.strip()
  return text


def tokenization(text):
  text = re.split('\W+', text)
  return text


stopword = nltk.corpus.stopwords.words('indonesian','english')

def stopwords(text):
  text = [word for word in text if word not in stopword]
  return text


from sklearn.pipeline import Pipeline
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

def stemming(text):
  factory = StemmerFactory()
  stemmer = factory.create_stemmer()
  do = []
  for w in text:
    dt = stemmer.stem(w)
    do.append(dt)
  d_clean=[]
  d_clean=" ".join(do)
  print(d_clean)
  return d_clean


def finalpreprocess(text):
  return stemming(stopwords(tokenization(preprocess(text))))
```

Lampiran 2. Class Multinomial Naïve Bayes

```
class MultinomialNB():

  def __init__(self, alpha=1):
    self.alpha = alpha

  def fit(self, X_train, y_train):
    m, n = X_train.shape
    self._classes = np.unique(y_train)
    n_classes = len(self._classes)

    # init: Prior & Likelihood
    self._priors = np.zeros(n_classes)
    self._likelihoods = np.zeros((n_classes, n))

    # Get Prior and Likelihood
    for idx, c in enumerate(self._classes):
        X_train_c = X_train[c == y_train]
        self._priors[idx] = X_train_c.shape[0] / m
        self._likelihoods[idx, :] = ((X_train_c.sum(axis=0))
+ self.alpha) / (np.sum(X_train_c.sum(axis=0) + self.alpha))


  def predict(self, X_test):
    return [self._predict(x_test) for x_test in X_test]

  def _predict(self, x_test):
    # Calculate posterior for each class
    posteriors = []
    for idx, c in enumerate(self._classes):
        prior_c = np.log(self._priors[idx])
        likelihoods_c = self.calc_likelihood(self._likelihood
s[idx,:], x_test)
        posteriors_c = np.sum(likelihoods_c) + prior_c
        posteriors.append(posteriors_c)

    return self._classes[np.argmax(posteriors)]
```

```
def calc_likelihood(self, cls_likeli, x_test):
   return np.log(cls_likeli) * x_test

 def score(self, X_test, y_test):
   y_pred = self.predict(X_test)
   return np.sum(y_pred == y_test)/len(y_test)
```

Lampiran 3. Class Support Vector Machine

```
def projection_simplex(v, z=1):
    """
    Projection onto the simplex:
        w^* = argmin_w 0.5 ||w-
v||^2 s.t. \sum_i w_i = z, w_i >= 0
    """
    n_features = v.shape[0]
    u = np.sort(v)[::-1]
    cssv = np.cumsum(u) - z
    ind = np.arange(n_features) + 1
    cond = u - cssv / ind > 0
    rho = ind[cond][-1]
    theta = cssv[cond][-1] / float(rho)
    w = np.maximum(v - theta, 0)
    return w

class MulticlassSVM(BaseEstimator, ClassifierMixin):

    def __init__(self, C=1, max_iter=50, tol=0.05,
                 random_state=None, verbose=0):
        self.C = C
        self.max_iter = max_iter
        self.tol = tol,
        self.random_state = random_state
        self.verbose = verbose

    def _partial_gradient(self, X, y, i):
        # Partial gradient for the ith sample.
        g = np.dot(X[i], self.coef_.T) + 1
        g[y[i]] -= 1
        return g

    def _violation(self, g, y, i):
        # Optimality violation for the ith sample.
        smallest = np.inf
        for k in range(g.shape[0]):
            if k == y[i] and self.dual_coef_[k, i] >= self.C:
```

```
                continue
            elif k != y[i] and self.dual_coef_[k, i] >= 0:
                continue
            smallest = min(smallest, g[k])
        return g.max() - smallest

    def _solve_subproblem(self, g, y, norms, i):
        # Prepare inputs to the projection.
        Ci = np.zeros(g.shape[0])
        Ci[y[i]] = self.C
        beta_hat = norms[i]*(Ci-self.dual_coef_[:,i])+g/norms[i]
        z = self.C * norms[i]
        # Compute projection onto the simplex.
        beta = projection_simplex(beta_hat, z)
        return Ci - self.dual_coef_[:, i] - beta / norms[i]

    def fit(self, X, y):
        n_samples, n_features = X.shape

        # Normalize labels.
        self._label_encoder = LabelEncoder()
        y = self._label_encoder.fit_transform(y)

        # Initialize primal and dual coefficients.
        n_classes = len(self._label_encoder.classes_)
        self.dual_coef_= np.zeros((n_classes, n_samples), dtype=n
p.float64)
        self.coef_ = np.zeros((n_classes, n_features))

        # Pre-compute norms.
        norms = np.sqrt(np.sum(X ** 2, axis=1))

        # Shuffle sample indices.
        rs = check_random_state(self.random_state)
        ind = np.arange(n_samples)
        rs.shuffle(ind)
```

```
violation_init = None
        for it in range(self.max_iter):
            violation_sum = 0
            for ii in range(n_samples):
                i = ind[ii]
                # All-zero samples can be safely ignored.
                if norms[i] == 0:
                    continue
                g = self._partial_gradient(X, y, i)
                v = self._violation(g, y, i)
                violation_sum += v
                if v < 1e-12:
                    continue
                # Solve subproblem for the ith sample.
                delta = self._solve_subproblem(g, y, norms, i)
                # Update primal and dual coefficients.
                self.coef_ += (delta * X[i][:, np.newaxis]).T
                self.dual_coef_[:, i] += delta

            if it == 0:
                violation_init = violation_sum
            vratio = violation_sum / violation_init

            if self.verbose >= 1:
                print("iter", it + 1, "violation", vratio)

            if vratio < self.tol:
                if self.verbose >= 1:
                    print("Converged")
                break
        return self

    def predict(self, X):
        decision = np.dot(X, self.coef_.T)
        pred = decision.argmax(axis=1)
        return self._label_encoder.inverse_transform(pred)
```

Lampiran 4. Class K-Nearest Neighbor

```
class KNN:
    def __init__(self, k):
        self.k = k

    def fit(self, X, y):
        self.X = X
        self.y = y

    def predict(self, X):
        y = np.zeros(len(X))
        # loop untuk semua data X baru
        for j, x in enumerate(X):
            sl = SortedList()
            # loop untuk data train
            for i, xt in enumerate(self.X):
                # hitung jarak
                dist = x - xt
                dist = dist.dot(dist)
                if len(sl) < self.k:
                    sl.add((dist, self.y[i]))
                else:
                    if dist < sl[-1][0]:
                        del sl[-1]
                        sl.add((dist, self.y[i]))
            # vote
            votes = {}
            for e in sl:
                if e[1] in votes.keys():
                    votes[e[1]] += 1
                else:
                    votes[e[1]] = 1

            # cari nilai max
            max_votes_class, max_votes = max(votes.items(), key=o
perator.itemgetter(1))
            y[j] = max_votes_class
        return y
```

```
def score(self, X, y):
        pred = self.predict(X)
        return np.mean(pred == y)
```

LEMBAR PERBAIKAN SKRIPSI

"PERBANDINGAN METODE NAÏVE BAYES, KNN DAN SVM PADA ANALISIS SENTIMEN JASA TRANSPORTASI ONLINE"

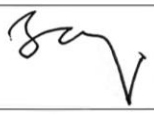
OLEH:

AYU ADHE PUTRI
D121181027

Skripsi ini telah dipertahankan pada Ujian Akhir Sarjana tanggal 12 April 2023.

Telah dilakukan perbaikan penulisan dan isi skripsi berdasarkan usulan dari penguji dan pembimbing skripsi.

Persetujuan perbaikan oleh tim penguji:

	Nama	Tanda Tangan
Ketua	Dr. Ir. Ingrid Nurtanio, M.T.	
Sekretaris	Dr. Amil Ahmad Ilham, S.T., M.IT.	
Anggota	Prof. Dr. Ir. Indrabayu, S.T., M.Bus.Sys., IPM, ASEAN .Eng.	
	Elly Warni. ST., M.T.	

Persetujuan perbaikan oleh pembimbing:

Pembimbing	Nama	Tanda Tangan
I	Dr. Ir. Ingrid Nurtanio, M.T.	
II	Dr. Amil Ahmad Ilham, S.T., M.IT.