LAMPIRAN – LAMPIRAN

1. Source Code Halaman Preprocessing Data

```
import re


# ============================
# CLEANING
# ============================
def cleaning(text):
    """
    Menghilangkan karakter selain huruf dan angka,
    serta mengubah teks menjadi huruf kecil.
    """
    text = re.sub(r'[^a-zA-Z0-9\s]', ' ', str(text))
    return text.lower().strip()
# ============================
# NORMALISASI
# ============================
kamus_normalisasi = {
    "gk":"tidak","ga":"tidak","ngga":"tidak","nggak":"tidak","kga":"tidak",
    "bgt":"banget","bngt":"banget","bgd":"banget","bgs":"bagus",
    "bener":"benar","skli":"sekali","sm":"sama","dr":"dari",
    "tp":"tapi","tpi":"tapi","yg":"yang","aja":"saja",
    "udh":"sudah","sdh":"sudah","uda":"sudah",
    "trs":"terus","trus":"terus","tmpt":"tempat",
    "jg":"juga","mntp":"mantap","mantul":"mantap",
    "km":"kamu","kmu":"kamu","jd":"jadi","krn":"karena"
}

def normalisasi(text):
    """
    Mengubah kata tidak baku menjadi kata baku
```

```
    berdasarkan kamus normalisasi.
    """
    words = text.split()
    return " ".join([kamus_normalisasi.get(w, w) for w in words])
# ============================
# TOKENIZING
# ============================
def tokenizing(text):
    """
    Memecah teks menjadi token kata.
    """
    return text.split()
# ============================
# STOPWORD REMOVAL
# ============================
stopwords = [
    "yang","dan","di","ke","dari","itu","ini",
    "karena","untuk","bagi","ada"
]

def remove_stopwords(tokens):
    """
    Menghapus kata-kata yang tidak memiliki
    makna penting dalam analisis sentimen.
    """
    return [w for w in tokens if w not in stopwords]
# ============================
# STEMMING
# ============================
def stemming(tokens):
    """
```

```
    Menghilangkan imbuhan awal dan akhir
    secara sederhana.
    """
    hasil = []
    for w in tokens:
        if w.startswith("mem"):
            w = w[3:]
        elif w.startswith("di") and len(w) > 3:
            w = w[2:]
        elif w.startswith("ke") and len(w) > 3:
            w = w[2:]
        elif w.endswith("nya"):
            w = w[:-3]
        hasil.append(w)
    return " ".join(hasil)
# ==============================
# PIPELINE PREPROCESSING
# ==============================
def preprocessing(text):
    """
    Pipeline preprocessing teks ulasan
    sebelum dilakukan klasifikasi sentimen.
    """
    text = cleaning(text)
    text = normalisasi(text)
    tokens = tokenizing(text)
    tokens = remove_stopwords(tokens)
    return stemming(tokens)
```

2. Source Code Pelatihan Model *Naïve Bayes*

```
@app.route('/naive_bayes', methods=['GET'])
```

```
def naive_bayes():
    if not session.get('admin'):
        return redirect(url_for('login'))

    # ======================
    # AKURASI
    # ======================
    acc = round(accuracy_score(y, y_pred) * 100, 2)

    # ======================
    # CLASSIFICATION REPORT
    # ======================
    report_raw = classification_report(
        y, y_pred, output_dict=True, zero_division=0
    )

    report = {
        k: {
            "precision": round(v["precision"], 2),
            "recall": round(v["recall"], 2),
            "f1-score": round(v["f1-score"], 2),
            "support": int(v["support"])
        }
        for k, v in report_raw.items()
        if k in ["positif", "negatif", "netral"]
    }

    # ======================
    # CONFUSION MATRIX
    # ======================
    labels = ["negatif", "netral", "positif"]
```

```
cm = confusion_matrix(y, y_pred, labels=labels)

cm_result = {
    "labels": labels,
    "matrix": cm.tolist()
}

# ========================
# SENTIMENT LABELING (DISTRIBUSI DATA)
# ========================
sentiment_labeling = (
    pd.Series(y)
    .value_counts()
    .reindex(labels, fill_value=0)
    .to_dict()
)

# ========================
# TF-IDF TOP KATA
# ========================
feature_names = vect.get_feature_names_out()
tfidf_mean = np.mean(X_tfidf.toarray(), axis=0)

top_idx = np.argsort(tfidf_mean)[-10:][::-1]

vect_df = [
    {
        "kata": feature_names[i],
        "tfidf": round(tfidf_mean[i], 4)
    }
    for i in top_idx
```

```
    ]

    return render_template(
        "naive_bayes.html",
        vect_df=vect_df,
        report=report,
        acc=acc,
        cm_result=cm_result,
        sentiment_labeling=sentiment_labeling,
        cm_image="confusion_matrix_nb.png",
        wordcloud_image="wordcloud_nb.png",
        error=None
    )
```

3. Source Code Prediksi Sentimen

```
@app.route('/prediksi', methods=['GET', 'POST'])
def prediksi():
    if not session.get('admin'):
        return redirect(url_for('login'))

    hasil = None
    error = None
    prob_result = None
    chart_image = None

    contoh_ulasan = (
        "Kampusnya sangat megah dan fasilitasnya lengkap, "
        "namun pelayanan administrasinya cukup lama"
    )

    if request.method == 'POST':
```

```
ulasan = request.form.get('ulasan')

if not ulasan:
    error = "Masukkan teks ulasan terlebih dahulu!"
else:
    try:
        # ========================
        # PREPROCESSING
        # ========================
        teks_bersih = preprocessing(ulasan)

        # ========================
        # TF-IDF TRANSFORM
        # ========================
        X_input = vect.transform([teks_bersih])

        # ========================
        # PROBABILITAS NAIVE BAYES
        # ========================
        proba = model.predict_proba(X_input)[0]
        classes = model.classes_

        prob_result = {
            classes[i]: round(proba[i] * 100, 2)
            for i in range(len(classes))
        }

        # ========================
        # SENTIMEN DOMINAN
        # ========================
        hasil = max(prob_result, key=prob_result.get)
```

```
            # ======================
            # GRAFIK BATANG DINAMIS
            # ======================
            plt.figure()
            plt.bar(prob_result.keys(), prob_result.values())
            plt.xlabel("Sentimen")
            plt.ylabel("Persentase (%)")
            plt.title("Distribusi Probabilitas Sentimen")

            chart_image = "prediksi_sentimen.png"
            plt.tight_layout()
            plt.savefig(os.path.join(STATIC_DIR, chart_image))
            plt.close()

        except Exception as e:
            error = f"Terjadi kesalahan: {str(e)}"

    return render_template(
        "prediksi.html",
        hasil=hasil,
        error=error,
        contoh_ulasan=contoh_ulasan,
        prob_result=prob_result,
        chart_image=chart_image
    )
```