

## DAFTAR GAMBAR

2.1	Arsitektur <i>Microservice</i> . . . . .	8
2.2	Arsitektur <i>Neural Network</i> . . . . .	9
3.1	Desain sistem . . . . .	12
3.2	Blok diagram sistem . . . . .	13
3.3	<i>Flowchart</i> . . . . .	15
3.4	Bentuk Data Mentah . . . . .	18
3.5	Bentuk Dataset .json . . . . .	19
3.6	Bentuk Dataset .csv . . . . .	19
3.7	Bentuk dataset sebelum dipetakan . . . . .	20
3.8	Bentuk dataset setelah dipetakan . . . . .	20
3.9	Bentuk dataset setelah diberi pembobotan . . . . .	21
3.10	Arsitektur <i>Neural Network</i> . . . . .	22
3.11	Hasil <i>Training</i> . . . . .	22
3.12	Tampilan Sebelum Prediksi . . . . .	50
3.13	Tampilan Setelah Prediksi . . . . .	50
4.1	Hasil Pengujian API dengan Aplikasi Insomnia . . . . .	53
4.2	Konfigurasi Jmeter untuk <i>Stress Test</i> . . . . .	54
4.3	Hasil <i>Stress Test</i> . . . . .	54
4.4	Hasil Pengujian Website Sebelum Prediksi . . . . .	55
4.5	Hasil Pengujian Website Setelah Prediksi . . . . .	56
4.6	Hasil Pengujian Prediksi Setiap Harinya . . . . .	57

```

class Prediksi(Resource):
    def get(self):

        formating = requests.get('http://127.0.0.1:3001/data')
        responnya = formating
        print("test")
        from tensorflow.keras.models import load_model
        new_model = load_model("mlmodel.h5")

        data = pd.concat(
            map(pd.read_csv, ['./Data/dataset.csv', './Data/daily_kab_bandung_14.csv']),
            ignore_index=True)
        print(type(data))
        label_encoder_column= ['reainFallTomorrow']

        encoder = LabelEncoder()
        for column in label_encoder_column:
            data[column] = encoder.fit_transform(data[column])

        def oneshoot_encoder(data, columns):
            for column in columns:
                dummies = pd.get_dummies(data[column])
                data = pd.concat([data, dummies], axis=1)
                data.drop(column, axis=1, inplace=True)
            return data

        data = oneshoot_encoder(data, ['windDirection'])
        y = data['reainFallTomorrow']

        dataTerakhir = data.iloc[-1]
        tempTanggal = dataTerakhir['tanggal']

        X = data.drop(['rainFallToday', 'tanggal', 'reainFallTomorrow'], axis=1)
        scaler = MinMaxScaler()
        X = pd.DataFrame(scaler.fit_transform(X), columns = X.columns)

        dfTest = pd.DataFrame(columns=list(X.columns))

        dfTest.to_csv("dataTest.csv", index=False)

        prediksi = new_model.predict(dfTest)
        prediksi = list(map(lambda x: np.argmax(x), prediksi))
        prediksi = np.array(prediksi)
        print(prediksi)

        tempTanggal2 = dataTerakhir['tanggal']
        arr_temp_tanggal2 = tempTanggal2.split('-')
        print(arr_temp_tanggal2)

        if(prediksi == 1):
            str_prediksi = 1
        else:
            str_prediksi = 0
        # Serialization
        numpyData = {
            "today": tempTanggal,
            "prediction": str_prediksi
        }
        encodedNumpyData = json.dumps(numpyData, cls=NumpyArrayEncoder) # use dump() to write array into
file
        data_pediksi = json.loads(encodedNumpyData)
        print(encodedNumpyData)

        return data_pediksi

    def post(self):
        print(request.json)
        return request.json

```