

APPLICATION OF ANN MODEL WITH BACKPROPAGATION ALGORITHM FOR PREDICTION LATE PAYMENT LOANS AT PNM MEKAAR

Muhammad Rizky Amra¹✉, Abdul Jabbar Lubis², Tengku Mohd. Diansyah³

^{1,2,3}Engineering and Computer, Harapan University Medan

✉e-mail: mrizky021201@gmail.com

Abstract

Study This aiming For overcome challenge the with building an *Artificial Neural Network* (ANN) model using algorithm *Backpropagation* For predict delay payment loans at PNM Mekaar branch Simalungun based on variables related or *features* that have been determined . ANN model with algorithm *Backpropagation* built own optimal performance . The model is built with milk *layer* (*hidden layer1* = 80 with activation *RELu* and *input layer* = 7, *hidden layer2* = 8 with *dropout* = 0.5 and activation *RELu* , *hidden layer3* = 8 with *dropout* = 0.3 and activation *RELu* , *hidden layer4* = 8 with *dropout* = 0.1 and activation *RELu* , *hidden layer5* = 1 with activation *Sigmoid*). The results of the model testing after evaluated with use *testing data* , obtained *score accuracy* reached 99.1%, the value *precision score* 100%, value *recall score* 98.9%, and *f1-score* by 99.4%. The model is *trained* use *batch size* parameters 64, *optimizer RMSprop* , *learning rate* 0.001, and number of *epoch* 100.

Say Key : *Prediction , Machine Learning, Artificial Neural Network, Backpropagation*

INTRODUCTION

Delay in pay installment loan No only cause Customer charged fines , but also have an impact on the collectibility status loans recorded in the System Service Information Finance Financial Services Authority (SLIK OJK) (Keniscayaan et al., 2020) . This Can make Customer experience difficulty moment want to submit future loans .

Loan can be one of source funding for society . Presence loan allow public For obtain the necessary funds For needs certain , such as to finance business . There are various reasons that can be cause delay in payments , such as condition economy that is not stable , management lack of finances good , and characteristics debtors which include nature , ability , capital, guarantee , and situation finance (Hasyim & Hasyim, 2024, Sary et al., 2023) . Research show that Lots Customer experience difficulty in fulfil obligation payment in accordance with the agreement that has been made .

PT Permodalan Nasional Madani (PNM) branch Simalungun is one of institution financial that provides service loan (credit) for society . At risk credit , delay payment loan is problem main in risk credit that can cause impact financial loss as well as bother operation business , especially PNM Mekaar branch Simalungun

With understand factors contributing to delays payment loans , PNM Mekaar branch Simalungun can

take steps proactive For reduce risk giving loans (credit) and increase quality portfolio loan company . Oh Because that , for overcome problem This is PNM Mekaar branch Simalungun can utilise progress technology in field intelligence Artificial Intelligence (*Artificial Intelligence*) Research This aiming For overcome challenge the with building an *Artificial Neural Network* (ANN) model using algorithm *Backpropagation* For predict delay payment loans at PNM Mekaar branch Simalungun . ANN model with Backpropagation algorithm has ability For analyzing large and complex data , as well Study from existing patterns in historical data (Saraf et al., 2023) . With using the data , this model can give more predictions accurate about possibility delay payment .

Development science in field *Artificial Intelligence* deliver variation technique best For do prediction (Noviani, 2023) . One of the frequently used algorithms used For training the ANN model is *Backpropagation* . In research (Agustina et al., 2022, Sary et al., 2023, Setiana et al., 2017) mention that *Backpropagation* is algorithm that can reduce mark *error* or *loss* so that make model accuracy can predict more accurate . Other research (Amaly et al., 2022) compare results performance ANN *Backpropagation* and ARMA methods for prediction inflation in Indonesia. Research results show that the most optimal method used For predict inflation For period furthermore is ANN *Backpropagation* method

Because own mark more *errors* small (Al-ayubi et al., 2020, Satria, 2020) . This matter become reference in choosing an ANN model with algorithm *Backpropagation* For do prediction delay payment loans at PNM Mekaar branch Simalungun

METHOD STUDY

Analysis system This aiming For give a better picture clear about the application that will built in study this , related with prediction delay payment loans at PNM Mekaar branch Simalungun . The application program that will built This is A the system that becomes tool help for PNM Mekaar party branch Simalungun For get convenience when do review candidate Customer or debtors who are in debt do submission loan .

Dataset used in study This is type of primary data obtained in a way directly from from sources that have been there is . Collection *dataset* obtained through interview to PNM Mekaar party branch Simalungun , in matter This with one of the *Senior Account Officer* staff (Mrs. Putri Rudiarti) for knowing the determination process eligibility loan to candidate debtor , obtain description criteria For determination eligibility loans and obtain data regarding category candidate debtor

Procedure Work system in study This explain about stages carried out in building a predictive model delay payment loans at PNM Mekaar Simalungun Branch with apply draft *Machine Learning* use *Artificial Neural Network* (ANN) with algorithm *Backpropagation* . The general diagram is from procedure Work system in study This can shown in the image under This .

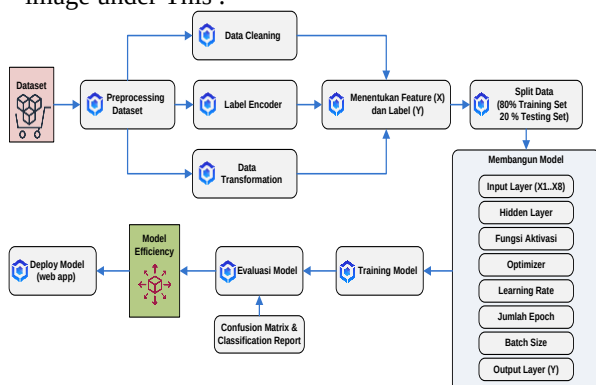


Figure 1 Procedure Diagram Work System

The image above show procedure Work system in building ANN models with algorithm

Backpropagation For predict delay payment loans at PNM Mekaar branch Simalungun .

Design system in study This done with make system *flowchart* and to design system interface . *Flowchart* system used For describe detailed process sequence and relationships between a process or Instructions with other processes in a program . *Flowchart* system in research This used For describe detailed process sequence and relationships between a process or Instructions with other processes in application predictions deployed into the A application based on *web* . As for *flowchart* system on the designed application can shown in the image .

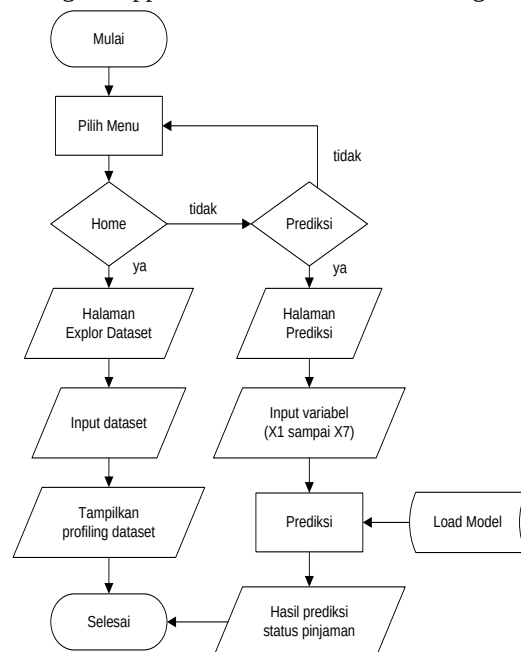


Figure 2 System Flowchart

The image above show show *flow chart* from the process flow in the application being built , starts with display page main . On the page Mainly , there are two menus available accessed by *the user* . If you select the ' *Home* ' menu so system will display page *profiling dataset* , where on the page This *user* can see *profiling* from *dataset* with method input *files dataset* with the .*xlsx* format . If the " *Prediction* " menu is selected , then system will display page prediction delay payment loan customers at PNM Mekaar branch Simalungun , where on the page This *user* moreover formerly input a number of variables to be made into input in determine the delay status payment loan customers (*debtors*) , then system will process and display results prediction regarding the delay status payment loan Customer with status information ' *Smooth* ' or ' *Stuck* ' .

Stage design *user interface* provide appearance interface For interaction between user with the system that will built . Design interface This It is expected that users can interact with easy in use system mentioned . Design interface in the application program This consists of of two designs interface , that is page main (*home*) for display *explore dataset* and pages prediction .

RESULTS AND DISCUSSION

Implementation system in study This explain about results training of Artificial Neural Network (ANN) models with algorithm *Backpropagation* For predict delay payment loan to PNM Mekaar . Model that has been trained Then will be in in evaluation use *confusion matrix* For know performance from the trained model , then do development (*deploy*) model into application based on web .

Model Training Results

When do *training* model, then determined a number of *hyperparameter* like type *optimizer* , value *learning rate* , amount *epoch* , and number *batch size* (Jatika et al., 2023) . Study This only done trials to a number of *hyperparameters* that have been determined with still refers to several results study previously . The *hyperparameters* tested were in study This can presented in the table below This :

Table 1Hyperparameter Training Model

No.	Parameter	Information
1.	<i>Optimizer</i>	<i>RMSprop</i>
2.	<i>Learning rate</i>	0.001
3.	<i>Batch Size</i>	64
4.	<i>Epoch</i>	100

The table above is *hyperparameters* to be tested on ANN models. Some *hyperparameters* used that is *optimizer* with *RMSprop* , value rate learning (*learning rate*) is 0.001, amount *batch size* namely 64, and the number *epoch* as many as 100. Concept deep model making study This with use *library* Hard . While Language programming used is *Python 3.10* with use help *Google Collaboratory* . As for the cut program code in ANN *Backpropagation Model* Creation can presented in the picture under This .

```

1 # Import library
2 from keras.models import Sequential
3 from keras.layers import Dense, Dropout
4 from keras.optimizers import RMSprop
5 from keras.utils import plot_model
6
7 # Fungsi untuk membuat model (menggunakan library Keras)
8 def create_model():
9     model = Sequential(name='My-Model-ANN')
10    model.add(Dense(80, input_dim=7, activation='relu', name='dense-layer-1'))
11    model.add(Dense(8, activation='relu', name='dense-layer-2'))
12    model.add(Dropout(0.5, name='dropout-layer-3'))
13    model.add(Dense(8, activation='relu', name='dense-layer-4'))
14    model.add(Dropout(0.5, name='dropout-layer-5'))
15    model.add(Dense(8, activation='relu', name='dense-layer-6'))
16    model.add(Dropout(0.1, name='dropout-layer-7'))
17    model.add(Dense(1, activation='sigmoid', name='dense-layer-8'))
18    opt_model = RMSprop(learning_rate=0.001)
19    model.compile(optimizer=opt_model, loss='binary_crossentropy', metrics=['accuracy'])
20    return model
21
22 # Create model
23 keras_model = create_model()
24
25 # Summary model
26 keras_model.summary()
27
28 # Plot model
29 plot_model(keras_model, to_file='My Model ANN.png', show_shapes=True,
30            show_layer_names=False, rankdir='TB', expand_nested=False, dpi=80)

```

Figure 3 Program Code for Create Model

Figure 3 is program code for make or *create* ANN model with use *library* *Tensorflow* and *Keras* as draft Model creation . The program code on lines 2 to 5 works For import a number of required *libraries* in ANN model creation . The program code in lines 8 to 20 is A *function* For create an ANN model. Next from *function* model that has been defined will be *created* with use The ' *create_model* ' command is listed in the program code line 23. The model that *has* been created can displayed *summary* - it use ' *summary* ' command listed on line 26 of the program code .

As for the results from *summary of* the ANN model that *has been* created previously can presented in Figure 4.

Model: "My-Model-ANN"		
Layer (type)	Output Shape	Param #
dense-layer-1 (Dense)	(None, 80)	640
dense-layer-2 (Dense)	(None, 8)	648
dropout-layer-3 (Dropout)	(None, 8)	0
dense-layer-4 (Dense)	(None, 8)	72
dropout-layer-5 (Dropout)	(None, 8)	0
dense-layer-6 (Dense)	(None, 8)	72
dropout-layer-7 (Dropout)	(None, 8)	0
dense-layer-8 (Dense)	(None, 1)	9
Total params: 1441 (5.63 KB)		
Trainable params: 1441 (5.63 KB)		
Non-trainable params: 0 (0.00 Byte)		

Figure 4Summary Model

Figure 4 is results from *summary of* the model

that has been *created* . Figure 3 is a program code for make or *create* For display results from the model plot can done use the '*plot_model*' command is listed in the program code lines 29 to 30. Figure 5 is results from the ANN model plot that has *been created* .

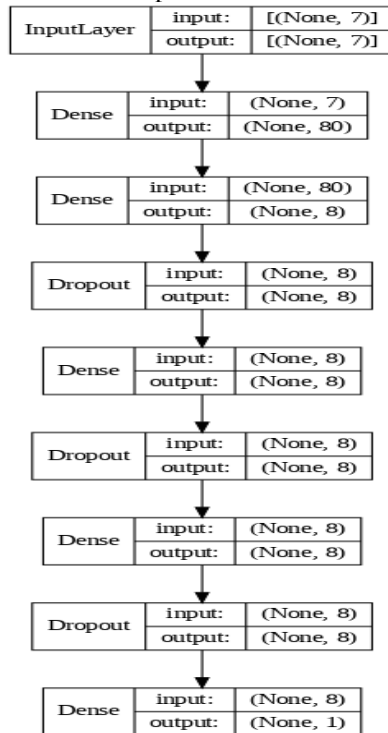


Figure 5 Model Plot Results

After finished build ANN model, then continued to the next stage *compile* the model then do *training* model. The program code for *compile* and *train* the model can presented in Figure 4.4.

```
1 # Build model menggunakan KerasClassifier
2 from scikeras.wrappers import KerasClassifier
3 keras_classifier = KerasClassifier(build_fn=keras_model, epochs=100,
4                                   batch_size=64, verbose=1,
5                                   callbacks=[HistoryCallback()],
6                                   validation_split=0.3)
```

Figure 6 Program Code for Compile Model

Figure 6 is The program code to compile the ANN model that has been *created* . The process of *compiling* the model with use *library KerasClassifier* . The program code in lines 2 to 6 works For operate order *compile* model. At the stage *compile model* is available a number of parameters used Among them , namely '*build=keras_model*' is a parameter for determines the model to be *compiled* . The parameter '*epoch s = 100*' is amount

iteration of the model to be *trained* , meaning will *training* process is carried out as many as 100 times. The parameter '*batch_size =64*' for *update* model weight at the time *training* model, then the parameter '*verbose=1*' works For display *progress bar* at the moment perform the model *training* process . The parameter '*callbacks = [HistoryCallback ()]*' serves For keep results *training* model which will be will used For visualize chart from results *training* model. While the parameter '*validation_split =0.3*' functions For allocate as much as 0.3 or 30% as validation data, which is taken from *training* data .

Compiled model Then will be *trained* with use the '*fit*' function as shown in figure 7.

```
1 # Buat alur (pipeline) dengan scaler (scaler menggunakan MinMax)
2 # dan model yang sudah didefinisikan sebelumnya
3 from sklearn.pipeline import Pipeline
4 from sklearn.preprocessing import MinMaxScaler
5 keras_model_pipeline = Pipeline([
6     ('scaler', MinMaxScaler(feature_range=(0, 1), copy=True, clip=False)),
7     ('classifier', keras_classifier)
8 ])
9
10 # Training model
11 keras_model_pipeline.fit(x_train, y_train)
```

Figure 7 Program Code for Training Model

Figure 7 is the program code used For do *training* ANN model. However before the model is *trained* , then moreover formerly make channel or *pipeline* from the ANN model with use '*Pipeline*' function contained in the *library sklearn* . At the time make *pipeline* model exists a number of parameters that must is specified , namely the '*sclaer*' parameter for determine function normalization *dataset* . In research This with use *min- maxscaler* as For normalization *dataset* . Next the '*classifier*' parameter for determine function classification that is with use *KerasClassifier* as stated in the program code lines 3 to 8. Stages next that is *training* model with use the '*fit*' function works For apply *model* on the *training* data (*x_train* , *y_train*) as stated in the program code line 11.

As for the results from the model *training* process ANN with algorithm *Backpropagation* can shown in figure 8.

```
10/10 [=====] - 2s 28ms/step - loss: 0.6538 - accuracy: 0.6650 - val_loss: 0.6273 - val_accuracy: 0.7963
Epoch 2/100
10/10 [=====] - 0s 6ms/step - loss: 0.5952 - accuracy: 0.7659 - val_loss: 0.5752 - val_accuracy: 0.7963
Epoch 3/100
10/10 [=====] - 0s 8ms/step - loss: 0.5909 - accuracy: 0.7914 - val_loss: 0.5418 - val_accuracy: 0.7963
Epoch 4/100
10/10 [=====] - 0s 7ms/step - loss: 0.5640 - accuracy: 0.7882 - val_loss: 0.5249 - val_accuracy: 0.7963
Epoch 5/100
10/10 [=====] - 0s 7ms/step - loss: 0.5542 - accuracy: 0.8010 - val_loss: 0.5166 - val_accuracy: 0.7963
Epoch 6/100
10/10 [=====] - 0s 7ms/step - loss: 0.5596 - accuracy: 0.8010 - val_loss: 0.5151 - val_accuracy: 0.7963
Epoch 7/100
10/10 [=====] - 0s 6ms/step - loss: 0.5486 - accuracy: 0.8025 - val_loss: 0.5126 - val_accuracy: 0.7963
Epoch 8/100
10/10 [=====] - 0s 9ms/step - loss: 0.5436 - accuracy: 0.8073 - val_loss: 0.5121 - val_accuracy: 0.7963
Epoch 9/100
10/10 [=====] - 0s 8ms/step - loss: 0.5215 - accuracy: 0.8057 - val_loss: 0.5076 - val_accuracy: 0.7963
Epoch 10/100
10/10 [=====] - 0s 7ms/step - loss: 0.5308 - accuracy: 0.8089 - val_loss: 0.5015 - val_accuracy: 0.7963
Epoch 11/100
10/10 [=====] - 0s 8ms/step - loss: 0.5103 - accuracy: 0.8073 - val_loss: 0.4971 - val_accuracy: 0.7963
Epoch 12/100
10/10 [=====] - 0s 7ms/step - loss: 0.5003 - accuracy: 0.8073 - val_loss: 0.4903 - val_accuracy: 0.7963
Epoch 13/100
10/10 [=====] - 0s 7ms/step - loss: 0.4994 - accuracy: 0.8073 - val_loss: 0.4836 - val_accuracy: 0.7963
Epoch 14/100
10/10 [=====] - 0s 8ms/step - loss: 0.5143 - accuracy: 0.8041 - val_loss: 0.4788 - val_accuracy: 0.7963
Epoch 15/100
10/10 [=====] - 0s 6ms/step - loss: 0.4873 - accuracy: 0.8073 - val_loss: 0.4746 - val_accuracy: 0.7963
```

Figure 8 Results of the Model Training Process

Figure 8 is visualization from the *training* process or *fitting* on the model, only displayed on *epochs* 1 to 15 of the total 100 *epochs* used . From the results of the model fitting process , will produced a number of mark among them mark *loss* state mark *loss* function from *training data* , value *accuracy* state mark *accuracy* from *training data* , value *val_loss* state mark *loss* function from *data validation* and *val_accuracy* state mark *accuracy* from *data validation* . At *epoch* First obtained mark *accuracy* of 0.7659 with mark *loss* of 0.5957. While *val_accuracy* reached 0.7963 with mark *val_loss* of 0.5752. At *epoch* First mark *val_accuracy* exceed mark *accuracy* , thing This to signify that the model is *trained* own good performance in do prediction delay payment loans to PNM Mekaar .

Result of *training model* Then will displayed in form chart with use library *matplotlib* as stated in program code in figure 9.

```
1 # Plot training and validation loss over epochs
2 import matplotlib.pyplot as plt
3
4 # Akses training history
5 history = pipeline.named_steps['classifier'].callbacks[0].history
6
7 # Plot training and validation loss
8 plt.figure(figsize=(15, 20))
9 plt.subplot(5, 2, 1)
10 plt.plot(np.arange(1, len(history['loss']) + 1), history['loss'],
11          label='Training Loss')
12 plt.plot(np.arange(1, len(history['val_loss']) + 1), history['val_loss'],
13          label='Validation Loss')
14 plt.title('Training and Validation Loss', fontsize=14, pad=10)
15 plt.xlabel('Epochs', fontsize=12)
16 plt.ylabel('Loss', fontsize=12)
17 plt.grid(alpha=0.2)
18 plt.legend()
19
20 # Plot training and validation accuracy
21 plt.subplot(5, 2, 2)
22 plt.plot(np.arange(1, len(history['accuracy']) + 1), history['accuracy'],
23          label='Training Accuracy')
24 plt.plot(np.arange(1, len(history['val_accuracy']) + 1), history['val_accuracy'],
25          label='Validation Accuracy')
26 plt.title('Training and Validation Accuracy', fontsize=14, pad=10)
27 plt.xlabel('Epochs', fontsize=12)
28 plt.ylabel('Accuracy', fontsize=12)
29 plt.grid(alpha=0.2)
30 plt.legend()
31
32 plt.tight_layout()
33 plt.show()
```

Figure 9 Program Code for Model Training Results Plot

Result of *training model* then will displayed in form graphs that can presented in Figure 10.

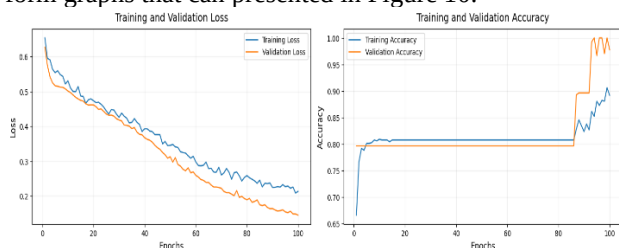


Figure 10 Accuracy and Loss Model graphs

Figure 10 parts adjacent left is mark from *training loss* and *validation loss* , while part adjacent right is mark from *training accuracy* and *validation accuracy* in *training* with amount *epoch* as many as 100. In the *epoch* First obtained mark *accuracy* of 0.7659 with mark *loss*

of 0.5957. While *val_accuracy* reached 0.7963 with mark *val_loss* of 0.5752. The *accuracy* and *val_accuracy* values Keep going experience increase until *epoch* 100. If measured between mark *accuracy* and *val_accuracy* show that the model does not experience *overfitting* . As for the *best epoch* from a total of 100 *epochs* used in *training* model namely is in *epoch* 99 with *accuracy* 0.9061 and *val_accuracy* 1.000 as well mark *loss* and *val_loss* Keep going down with a loss of 0.2084 and a *val_loss* of 0.1484.

Model Evaluation Results

At the stage This will discuss about evaluation performance of the trained model using independent test data . Measuring metric evaluation like *accuracy score*, *precision score*, *recall score* , and *f1-score* with use *confusion matrix* , for evaluate how much both models that have been trained in do prediction delay payment loans at PNM Mekaar . Program code for display *confusion matrix* can presented in Figure 11.

```
1 # Plot Confusion Matrix
2 from sklearn.metrics import confusion_matrix
3 import seaborn as sns
4 label = ['Lancar', 'Macet']
5 prob_prediksi = keras_model.pipeline.predict(x_test)
6 cm_model = confusion_matrix(y_test, prob_prediksi)
7 plt.figure(figsize=(6, 3))
8 sns.heatmap(cm_model, annot=True, fmt='.4g', linewidths=2, cmap='YlGnBu',
9            cbar=True, xticklabels=label, yticklabels=label)
10 plt.xlabel('Predicted Class', fontsize=12, color='darkblue', labelpad=20)
11 plt.ylabel('True Class', fontsize=12, color='darkblue', labelpad=20)
12 plt.title('Confusion Matrix', fontsize=12, pad=20)
13 plt.show()
```

Figure 11 Program Code Displaying Confusion Matrix Model

Figure 11 is the program code used For display results confusion matrix of the model that has been tested with use *data testing* . The results are as follows model evaluation using *confusion matrix* being tested use *data testing* can presented in Figure 12.

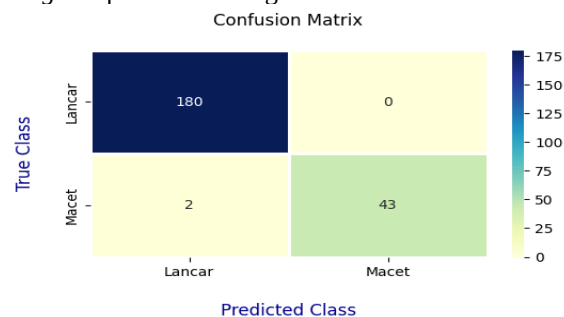


Figure 12 Confusion Matrix Model Results

Figure 12 is results *confusion matrix* from evaluation performance of the trained model using independent test data . From the results *confusion matrix* can known model performance with use *classification*

report . The program code for display results *classification report* can presented in Figure 13.

```

1 # Print Classification Report
2 from sklearn.metrics import classification_report
3 label = ['Lancar', 'Macet']
4 prob_prediksi = keras_model_pipeline.predict(x_test)
5
6 cr_model = classification_report(y_test, prob_pred,
7                                 target_names=label)
8 print(cr_model)

```

Figure 13 Program Code Displaying Classification Report Model Results

Based on Figure 12 and Figure 13, information from *confusion matrix* used For calculate the parameters of accuracy , *precision* , *recall* , and *f1 - score* from the model, but moreover formerly done search mark from *True Positive* (TP), *False Positive* (FP), *False Negative* (FN), and *True Negative* (TN) of each *label / class* . The meaning of TP, FP, FN, and TN is as following :

TP : *predicted class* “ Current ” → *true class* “ Current ”

FP : *predicted class* “ Current ” → *true class* “ other than Congested ”

FN : *predicted class* “ Jammed ” → *true class* “ Jammed ”

TN : *predicted class* “ Congestion ” → *true class* “ Smooth ”

Based on Figure 4.10, then can known The total number of *True Positive* (TP) was 180, the total number of *False Positive* (FP) was 0, the total number of *False Negative* (FN) was 2, and the total number of *True Negative* (TN) was 43, where mark This will used For calculate the parameters of *accuracy score* , *precision score* , *recall score* , and *f1-score* . The following is explanation from results calculation for each parameter:

1. *Accuracy score* is comparison between amount correct prediction with amount overall prediction right and wrong. As for the results *accuracy model* scenario after testing using *data testing* can counted use formula in equation (2.17).

$$\begin{aligned}
 \text{Accuracy Score} &= \frac{TP + TN}{TP + FP + FN + TN} \\
 &= \frac{180 + 43}{180 + 0 + 2 + 43} \\
 &= \frac{223}{225} = 0,991
 \end{aligned}$$

2. *Precision Score* , describes accuracy between the requested data with results model prediction . The results are *precision model* after testing with *data testing* can counted use formula in equation (2.18).

$$\text{Precision Score} = \frac{TP}{TP + FP} = \frac{180}{180 + 0} = 1,000$$

3. *Recall Score* , describes success of the model in find

return A information . As for the results *recall model* after testing with *data testing* can counted use formula in equation (2.19).

$$\text{Recall Score} = \frac{TP}{TP + FN} = \frac{180}{180 + 2} = \frac{180}{182} = 0,989$$

4. *F1-score* , describes comparison of the average *precision* and weighted *recall* . The results are *f1-score model* after testing with *data testing* can counted use formula in equation (2.20).

$$\begin{aligned}
 f1score &= 2x \frac{\text{recall} \times \text{precision}}{\text{recall} + \text{precision}} \\
 &= 2 \left(\frac{0,989 \times 1,000}{0,989 + 1,000} \right) = 0,994
 \end{aligned}$$

Based on results calculation above , then can taken conclusion that *score accuracy* from the model after being tested using *data testing* reached 0.991 (99.1%), the value *precision score* 1,000 (100%), value *recall score* 0.989 (98.9%), and *f1-score* of 0.994 (99.4%). The test results with use *data testing* as many as 225 data with details 180 for ' Current ' category and 45 for ' Jam ' category . Test results show that For category ' Current ' , all data can be predicted with accurate by the model. While For ' Congestion ' category , there are 43 predicted data with correct and the remaining 2 predicted incorrectly by the model.

Testing System

Model that *has* been trained Then will be *deployed* into the A application based on *web* as a medium for make it easier user (*user*) does prediction delay payment loans to PNM Mekaar who want known . The process of *deploying* the model into application with use *Streamlit* as *library main* . Testing done with objective For know that the application program is created in accordance with results analysis and design as well as produce One conclusion is the program created in accordance with what is expected and successful operate its functions with true . The application developed (*deployed*) consists of from appearance page main (*home*) and pages predictions . As for the results testing on page *home* can shown in figure 14.

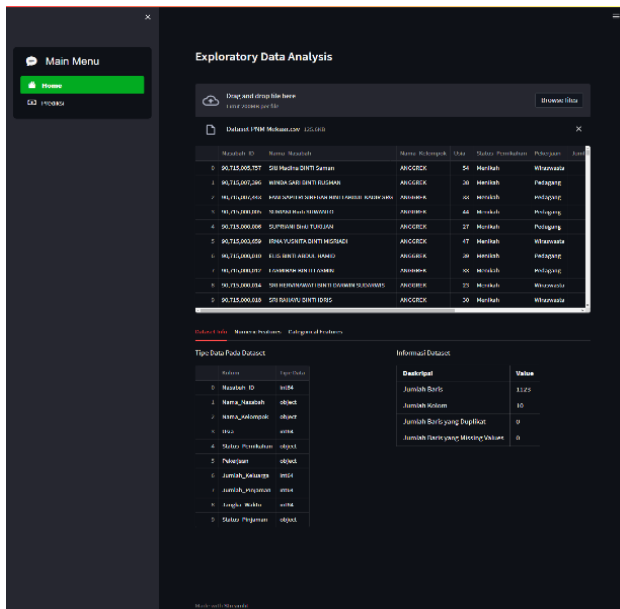


Figure 14 Home Page Test Results

Figure 14 is results testing on the home page , where on the page This system will display information about *dataset* from PNM Mekaar customers . The results testing from deployed applications in do prediction with category congested can shown in figure 15.

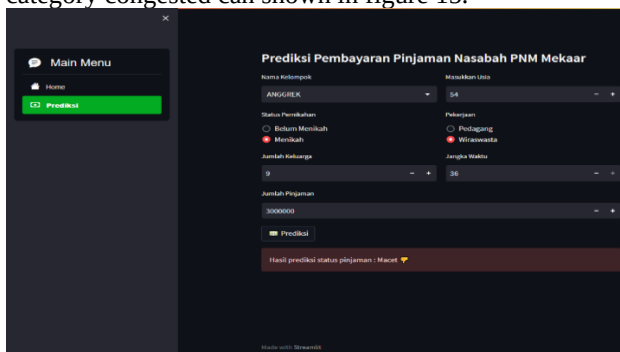


Figure 15 Prediction Result Testing With Loan Status Congested

Figure 15 shows results testing from deployed applications in perform the prediction process delay payment loan to PNM Mekaar . System succeed do prediction with Correct in accordance with category or label of The *testing data* used . The results are as follows testing from deployed applications in do prediction with category fluent can shown in figure 16.

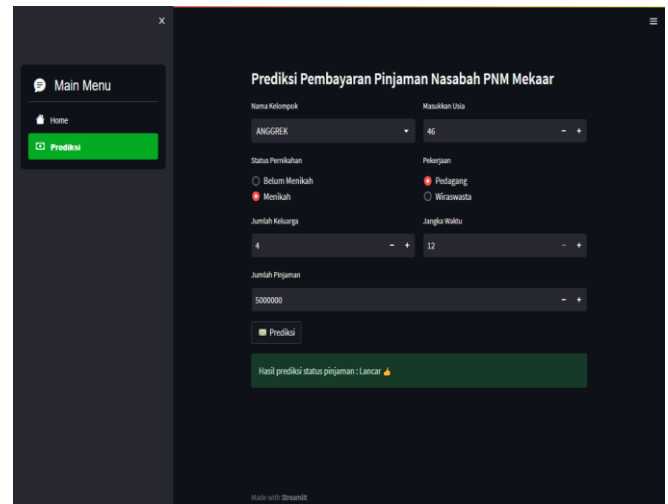


Figure 16 Prediction Result Testing With Loan Status Fluent

Figure 16 shows results testing system with loan status smooth , system succeed do prediction with Correct in accordance with category the *testing data* used.

CONCLUSION

ANN model with algorithm *Backpropagation* built based on results experiment For reach optimal performance consists of from 1 *input layer* with 7 units *neurons* , 4 *hidden layers* (80 *neuron units* in the 1st *hidden layer* , 8 *neuron units* in the 2nd *hidden layer* , 8 *neuron units* in the 3rd *hidden layer* , and 8 *neuron units* in the 4th *hidden layer* with function activation *RELU* For every *hidden layer*) and 1 *neuron* in the *output layer* with function activation *Sigmoid* . The parameters used For train the built model , including *learning rate* with value 0.001, amount *batch size* 64, type *RMSprop optimizer* , and the number of *epoch* 100.

2. The level of accuracy of the ANN model with algorithm *backpropagation* in study This show satisfactory result with *score accuracy* reached 0.991 (99.1%) . This model capable predict delay payment loan with high accuracy , indicating that the model is built can reliable For needs prediction .

The performance of the ANN model can be evaluated use *confusion matrix* For evaluate model capabilities in predict delay payment loan with right . Based on results *confusion matrix* obtained metric evaluation with *accuracy score* 0.991 (99.1%) , *precision* 1,000 (100%) , *recall score* 0.989 (98.9%) , and *f1-score* 0.994 (99.4%) which is adequate . This result show that the model is built after evaluation capable differentiate between customers who are late (*traffic jams*) and not late (*smooth*) payment loan with level optimal accuracy and level minimal errors .

ANN model that has been built can be implemented to in A application based on *web* , so that can make it easier PNM Mekaar party in predict delay payment loan . Application This allow user For enter customer data and obtain results prediction in a way fast and accurate .

Distribution *dataset* in training the model is done with the optimal percentage , namely 80% for training data (*training data*) and 20% for testing data (*testing data*). The division *dataset* ensure that the model has enough data for learn and be tested , so that capable give good performance on unprocessed data Once seen previously .

REFERENCE

- Agustina, I., Mulyani, Y., & Septiana, T. (2022). ANALYSIS OF KICKSTARTER SUCCESS PREDICTION MODEL DEVELOPMENT USING BACKPROPAGATION AND RANDOM FOREST ALGORITHMS. *Journal of Informatics and Applied Electrical Engineering* , 10 (3).
- Al-ayubi, S., Novianty, A., & Prasasti, MTAL (2020). ESTIMATION OF EARTHQUAKE MAGNITUDE FROM P WAVE SEISMIC SIGNAL USING BACKPROPAGATION ANN METHOD. In *Proceeding of Engineering* (Vol. 7, pp. 4624–4632).
- Amaly, H., Haiban, R., & Basirun, H. (2022). COMPARISON OF ANN BACKPROPAGATION AND ARMA METHODS FOR INFLATION FORECASTING IN INDONESIA. *JOURNAL OF PROBABILITY AND STATISTICS* , 3 (November).
- Hasyim, U., & Hasyim, U. (2024). Analysis of Factors Affecting Late Payments in the Financial Management of BMT Muamalah Syariah Tebuireng. *Scientific Journal of Economics and Management* , 2 (9), 97–107.
- Jatika, P.L., Rizki, F., Pajar, M., Putra, K., Assuja, M.A., & Ariany, F. (2023). Implementation of Deep Learning Lenet with Data Augmentation on Orchid Identification. *JOURNAL OF INFORMATICS AND SOFTWARE ENGINEERING* , 4 (September), 357–366.
- Keniscayaan, S., Era, DI, & Dan, D. (2020). Concept and Comparison of Buy Now, Pay Later with Bank Credit in Indonesia: An Inevitability in the Digital and Technology Era. *Jurnal Rechts Vinding* , 9 (2), 183–201.
- Noviani, D. (2023). Prediction of Drug Usage Quantity in Health Services Using Backpropagation Neural Network Algorithm. *IHSANIKA: Journal of Islamic Religious Education* , 1 (4).
- Saraf, J., Jst, T., Saraf, J., Jst, T., Hartanto, A., & Tarigan, CN (2023). Forecasting the Consumer Price Index (CPI) of Medan City Using Backpropagation Artificial Neural Networks (ANN). In *Series: Energy & Engineering* (Vol. 6, pp. 0–6). <https://doi.org/10.32734/ee.v6i1.1799>
- Sary, MS, Mas'ud, R., & Azkar, M. (2023). PT Bank NTB Syariah's Strategy in Overcoming Problematic Musyarakah Financing During the Covid-19 Period. *Journal of Economics & Sharia Economics* , 6 (2), 1223–1235.
- Satria, W. (2020). BACKPROPAGATION ARTIFICIAL NEURAL NETWORK FOR PRODUCT SALES FORECASTING (CASE STUDY AT METRO ELECTRONIC AND FURNITURE). *Journal of Information Technology Research* , 1 (1), 14–19.
- Setiana, M., Rahayudi, B., & Muflikhah, L. (2017). Backpropagation Method to Predict Open Unemployment Rate (TPT) in West Java Province with Adam Optimization. *Journal of Information Technology Development and Computer Science* , 1 (1), 1–10.