Daftar Pustaka

Akhmad Hizham, F., Nurdiansyah, Y., & Media Firmansyah, D. (n.d.-a). *Hizham et al., Implementasi Metode Backpropagation Neural Network (BNN) Implementasi Metode Backpropagation Neural Network (BNN) dalam Sistem Klasifikasi Ketepatan Waktu Kelulusan Mahasiswa (Studi Kasus: Program Studi Sistem Informasi Universitas Jember) (Implementation of Backpropagation Neural Network (BNN) Method in Classification System of Timeliness of Graduation Students (Case Study: Information System Study Program of Jember University))*.

Akhmad Hizham, F., Nurdiansyah, Y., & Media Firmansyah, D. (n.d.-b). *Hizham et al., Implementasi Metode Backpropagation Neural Network (BNN) Implementasi Metode Backpropagation Neural Network (BNN) dalam Sistem Klasifikasi Ketepatan Waktu Kelulusan Mahasiswa (Studi Kasus: Program Studi Sistem Informasi Universitas Jember) (Implementation of Backpropagation Neural Network (BNN) Method in Classification System of Timeliness of Graduation Students (Case Study: Information System Study Program of Jember University))*.

Andrijasa, M. (2010). Mistianingsih, dan, kunci. *K., Pengangguran, P., Algoritma Backpropagation, Dan, Teknologi Informasi, J., & Negeri Samarinda, P*.

Awalluddin, M. A., Tuan Nooriani, T. I., & Maznorbalia, A. S. (2022). THE RELATIONSHIP BETWEEN PERCEIVED PRESSURE, PERCEIVED OPPORTUNITY, PERCEIVED RATIONALIZATION AND FRAUD TENDENCY AMONG EMPLOYEES: A STUDY FROM THE PEOPLE'S TRUST IN MALAYSIA. *Studies in Business and Economics*, *17*(2), 23–43. https://doi.org/10.2478/sbe-2022-0023

Bisnis dan Ekonomi, J., Suzana Tina Lestari Program Studi Akuntansi, S., Tinggi Ilmu Ekonomi Pancasetia Banjarmasin Jl Yani Km, S. A., & Selatan, K. (2019). Penyebab Fraud pada Perusahaan di Kota Banjarmasin. In *JBE* (Vol. 26, Issue 2). https://www.unisbank.ac.id/ojs;

Breiman, L., & Cutler, A. (2005). *Random Forests. Berkeley*.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining*, 785–794.

De Diego, I. M., Redondo, A. R., Fernández, R. R., Navarro, J., & Moguerza, J. M. (2022). General Performance Score for classification problems. *Applied Intelligence*, *52*(10), 12049–12063. https://doi.org/10.1007/s10489-021-03041-7

Deng, X., Liu, Q., Deng, Y., & Mahadevan, S. (2016). An improved method to construct basic probability assignment based on the confusion matrix for classification problem. *Information Sciences*, *340–341*, 250–261. https://doi.org/10.1016/j.ins.2016.01.033

Eldin Mohammed Abd El-Hamid Ahmed Abdou, H., Khalifa, W., Ismail Roushdy Professor, M., Salem, A.-B. M., El-Hamid, A., Eldin Mohammed Ahmed Abdou, H., Ismail, M., Eldin Abd Elhamid, H. M., & Roushdy, M. (2019). Machine Learning Techniques for Credit Card Fraud Detection Machine Learning Techniques for Credit Card Fraud Detection Recommended Citation Recommended Citation Machine Learning Techniques for Credit Card Fraud Detection. In *Future Computing and Informatics Journal* (Vol. 4, Issue 2). https://digitalcommons.aaru.edu.jo/fcij/vol4/iss2/5

Estévez, P. A., Held, C. M., & Perez, C. A. (2006). Subscription fraud prevention in telecommunications using fuzzy rules and neural networks. *Expert Systems with Applications*, *31*(2), 337–344. https://doi.org/10.1016/j.eswa.2005.09.028

Gajbhiye, S. (2013). Next Generation Revenue Assurance. *Telecom Business Review*, *6*(1), 34.

Gonçalves, L., Subtil, A., Oliveira, M. R., De, P., & Bermudez, Z. (2014). ROC CURVE ESTIMATION: AN OVERVIEW. In *REVSTAT-Statistical Journal* (Vol. 12, Issue 1).

Hastie, T., Tibshirani, R., & Friedman, J. (2009). Random forests. In *The elements of statistical learning* (pp. 587–604). Springer.

Huang, J. C., Tsai, Y. C., Wu, P. Y., Lien, Y. H., Chien, C. Y., Kuo, C. F., Hung, J. F., Chen, S. C., & Kuo, C. H. (2020). Predictive modeling of blood pressure during hemodialysis: a comparison of linear model, random forest, support vector regression, XGBoost, LASSO regression and ensemble method. *Computer Methods and Programs in Biomedicine*, *195*. https://doi.org/10.1016/j.cmpb.2020.105536

Jain, V. (2017). Perspective analysis of telecommunication fraud detection using data stream analytics and neural network classification based data mining. *International Journal of Information Technology (Singapore)*, *9*(3), 303–310. https://doi.org/10.1007/s41870-017-0036-5

JEPIN (Jurnal Edukasi dan Penelitian Informatika) Peningkatan Kinerja Akurasi Prediksi Penyakit Diabetes Mellitus Menggunakan Metode Grid Seacrh pada Algoritma Logistic Regression. (n.d.-a).

JEPIN (Jurnal Edukasi dan Penelitian Informatika) Peningkatan Kinerja Akurasi Prediksi Penyakit Diabetes Mellitus Menggunakan Metode Grid Seacrh pada Algoritma Logistic Regression. (n.d.-b).

Kabari, L. G. (2016). Telecommunications Subscription Fraud Detection Using Naïve Bayesian Network. In *International Journal of Computer Science and Mathematical Theory* (Vol. 2, Issue 2). www.iiardpub.org

Ketua Umum APJII MUHAMMAD ARIF. (n.d.).

Khatri, S., Arora, A., & Prakash Agrawal, A. (n.d.). *Supervised Machine Learning Algorithms fo r Credit Card Fraud Detection: A Comparison*.

Kou, Y., Lu, C.-T., Sinvongwattana, S., & Huang, Y.-P. (n.d.). *Survey of Fraud Detection Techniques*.

Kryvinska, N., & Greguš, M. (n.d.). *Studies in Systems, Decision and Control 330*. http://www.springer.com/series/13304

Krzanowski, W. J., & Hand, D. J. (2009). *ROC curves for continuous data*. Chapman and Hall/CRC.

Lu, N., Lin, H., Lu, J., & Zhang, G. (2014). A customer churn prediction model in telecom industry using boosting. *IEEE Transactions on Industrial Informatics*, *10*(2), 1659–1665. https://doi.org/10.1109/TII.2012.2224355

Lu, S., Li, Q., Bai, L., & Wang, R. (2019). Performance predictions of ground source heat pump system based on random forest and back propagation neural network models. *Energy Conversion and Management*, *197*. https://doi.org/10.1016/j.enconman.2019.111864

Martinez-Plumed, F., Contreras-Ochando, L., Ferri, C., Hernandez-Orallo, J., Kull, M., Lachiche, N., Ramirez-Quintana, M. J., & Flach, P. (2021). CRISP-DM Twenty Years Later: From Data Mining Processes to Data Science Trajectories. *IEEE Transactions on Knowledge and Data Engineering*, *33*(8), 3048–3061. https://doi.org/10.1109/TKDE.2019.2962680

Mattison, R. (2005). *The telco revenue assurance handbook*. Lulu. com.

Meng, C., Zhou, L., & Liu, B. (2020). A case study in credit fraud detection with SMOTE and XGboost. *Journal of Physics: Conference Series*, *1601*(5). https://doi.org/10.1088/1742-6596/1601/5/052016

PENERAPAN ANALISIS RANDOM FOREST PADA PROTOTYPE SISTEM PREDIKSI HARGA KAMERA BEKAS MENGGUNAKAN FLASK. (n.d.).

Qiu, Y., Zhou, J., Khandelwal, M., Yang, H., Yang, P., & Li, C. (2021). Performance evaluation of hybrid WOA-XGBoost, GWO-XGBoost and BO-XGBoost models to predict blast-induced ground vibration. *Engineering with Computers*. https://doi.org/10.1007/s00366-021-01393-9

Rosset, S., Murad, U., Neumann, E., Idan, Y., & Pinkas, G. (1999). *Discovery of Fraud Rules for Telecommunications-Challenges and Solutions*.

SCOPE AND NATURE OF INTERNET FRAUD. (n.d.). www.traderlist.com

Shinde, P. P., & Shah, S. (2018). A review of machine learning and deep learning applications. *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, 1–6.

Shing Lim, K., Hong Lee, L., & Sim, Y.-W. (2021). A Review of Machine Learning Algorithms for Fraud Detection in Credit Card Transaction. *IJCSNS International Journal of Computer Science and Network Security*, *21*(9). https://doi.org/10.22937/IJCSNS.2021.21.9.4

Somvanshi, M., & Chavan, P. (n.d.). *A Review of Machine Learning Techniques using Decision Tree and Support Vector Machine*.

Stewart, D. (2006). *Measuring Execution Time and Real-Time Performance*.

Streiner, D. L., & CAIRNEY, J. (2013). 24 What's under the ROC? An Introduction to Receiver Operating Characteristic Curves. *A Guide for the Statistically Perplexed: Selected Readings for Clinical Researchers*, *52*, 304.

Swetha, P., & Dayananda, R. B. (2020). Improvised_XgBoost Machine learning Algorithm for Customer Churn Prediction. *EAI Endorsed Transactions on Energy Web*, *7*(30), e14–e14.

Trigila, A., Iadanza, C., Esposito, C., & Scarascia-Mugnozza, G. (2015). Comparison of Logistic Regression and Random Forests techniques for shallow landslide susceptibility assessment in Giampilieri (NE Sicily, Italy). *Geomorphology*, *249*, 119–136. https://doi.org/10.1016/j.geomorph.2015.06.001

Venkata Suryanarayana, S., Balaji, G. N., & Venkateswara Rao, G. (2018). Machine learning approaches for credit card fraud detection. *International Journal of Engineering and Technology(UAE)*, *7*(2), 917–920. https://doi.org/10.14419/ijet.v7i2.9356

Vitor, J., & De Sousa, C. (2014). *Telecommunication Fraud Detection Using Data Mining techniques*.

wardhani. (n.d.).

Weiss, G. M. (2009). *Data Mining in the Telecommunications Industry Section: Service Data Mining in the Telecommunications Industry*. https://www.researchgate.net/publication/251741570

LAMPIRAN

Lampiran 1 Model Raw Data

NO	IS_FRAUD	DATA CUTOFF	NCLI	ND_INET	REG	WITEL	DATEL	STO	CITEM_SPEED	SPEED_INET	STATUS_INET	USAGE	BILLING	UMUR_LANGGANAN	PRA_PASCA	FLAGGING
13314	0	202012	34709406		7	MAKASSAR		TMA	INETF100M	100M	active	1605.86	943500	52	PASCABAYAR	
14754	0	202012	54168691		7	MAKASSAR		BAL	INETC10M	10M	active	335.35	284200	30	PASCABAYAR	
130911	0	202012	53227060		7	MAKASSAR		ANT	INETC20M	20M	active	447.72	384700	56	PASCABAYAR	
102861	0	202012	55857455		7	MAKASSAR		PNK	INETC40M	40M	active	883.64	541100	27	PASCABAYAR	
1296	1	202012	54576743		7	MAKASSAR		BAL	INETC10M	10M	active	1171.97	202000	34	PASCABAYAR	fraud
58981	0	202012	50196628		7	MAKASSAR		BAL	INETC10M	10M	active	445.19	207000	52	PASCABAYAR	

40152	0	202012	45035518		7	MAKASSAR		KIM	INETC10M	10M	active	306.67	268300	55	PASCABAYAR	
125733	0	202012	49686636		7	MAKASSAR		PKN	INETC10M	10M	active	356.44	239700	30	PASCABAYAR	
15774	0	202012	36017526		7	MAKASSAR		SUG	INETC20M	20M	active	699.58	419400	19	PASCABAYAR	
30356	0	202012	30417543		7	MAKASSAR		TMA	INETC50M	50M	active	678.43	766300	11	PASCABAYAR	
3718	0	202012	38745721		7	MAKASSAR		SUG	INETC10M	10M	active	226.74	235800	1	PASCABAYAR	
37798	0	202012	50130501		7	MAKASSAR		BAL	INETC20M	20M	active	408.98	422500	27	PASCABAYAR	
1904	0	202012	54538426		7	MAKASSAR		WTP	INETC10M	10M	active	186.13	171500	15	PASCABAYAR	
64202	0	202012	40631302		7	MAKASSAR		PNK	INETC20M	20M	active	714.58	472100	53	PASCABAYAR	

21706	0	202012	42691575		7	MAKASSAR		MAT	INETC 40M	40M	active	531.47	533900	47	PASCABAYAR	
89847	0	202012	51956786		7	MAKASSAR		WTP	INETC 50M	50M	active	732.46	692300	40	PRABAYAR	
16515	0	202012	40464230		7	MAKASSAR		MAR	INETC 40M	40M	active	1060.67	554000	49	PASCABAYAR	
54035	0	202012	45012548		7	MAKASSAR		SUD	INETC 40M	40M	active	624.18	540400	30	PASCABAYAR	
2084	0	202012	43671131		7	MAKASSAR		WTP	INETC 10M	10M	active	329.76	266400	31	PASCABAYAR	
120590	0	202012	47046979		7	MAKASSAR		PNK	INETC 10M	10M	active	336.93	213600	37	PASCABAYAR	
126827	0	202012	59702969		7	MAKASSAR		SUG	INETC 40M	40M	active	450.07	529400	52	PASCABAYAR	

46922	0	202012	39725088		7	MAKASSAR		ANT	INETC 40M	40M	active	1059.68	529200	11	PASCABAYAR	
84757	0	202012	51769740		7	MAKASSAR		MAT	INETC 40M	40M	active	733.19	540500	14	PASCABAYAR	
78022	0	202012	31981862		7	MAKASSAR		SUD	INETC 10M	10M	active	315.85	220000	1	PASCABAYAR	
58331	0	202012	30433182		7	MAKASSAR		SIN	INETC 10M	10M	active	446.68	190700	23	PASCABAYAR	
81338	0	202012	33428306		7	MAKASSAR		SUD	INETC 10M	10M	active	435.97	186400	7	PASCABAYAR	
59142	0	202012	49642873		7	MAKASSAR		BAL	INETC 20M	20M	active	714.5	501600	6	PASCABAYAR	
130405	0	202012	40376559		7	MAKASSAR		ANT	INETC 20M	20M	active	251.28	430100	59	PASCABAYAR	

27571	0	202012	36011368		7	MAKASSAR		KIM	INETC40M	40M	active	1034.66	540400	21	PASCABAYAR	
27602	0	202012	45210913		7	MAKASSAR		SUG	INETC10M	10M	active	201.52	270700	40	PASCABAYAR	
90757	0	202012	55602164		7	MAKASSAR		SUD	INETF50M	50M	active	691.64	685500	5	PASCABAYAR	
14335	0	202012	49782164		7	MAKASSAR		BAL	INETC200M	200M	active	2646.49	1859400	39	PASCABAYAR	
104338	0	202012	47353067		7	MAKASSAR		BAL	INETF20M	20M	active	499.29	447200	59	PASCABAYAR	
104782	0	202012	31845393		7	MAKASSAR		BAL	INETC50M	50M	active	1324.78	800900	8	PASCABAYAR	

Lampiran 2 Package yang digunakan

```
from google.colab import drive
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import xgboost as xgb
from sklearn import preprocessing
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
# Decision tree algorithm
from sklearn.model_selection import train_test_split
# Data Splitting
from sklearn.preprocessing  import StandardScaler
# Data Standadization
from sklearn.preprocessing import MinMaxScaler
#Min-Max Data Normalization
from sklearn.ensemble import RandomForestClassifier
# Random Forest tree algorithm
from sklearn.linear_model import LogisticRegression
# Logistic regression algorithm
from imblearn.over_sampling import ADASYN
# Oversampling Data with ADASYN
```

Lampiran 3 Import Data

```
#IMPORTING DATA
url = 'https://raw.githubusercontent.com/AdyAhmadi24/Fraud/main/Flagging_Witel_202012.csv'
df = pd.read_csv(url)

#Check the columns name
df.columns.values.tolist()
```

Lampiran 4 Preprocessing data

```
#Removing unnecessary column
uc = ['IS_FRAUD', 'NO', 'DATA CUTOFF', 'NCLI', 'ND_INET', 'REG', 'WITEL',
 'DATEL',  'CITEM_SPEED','STATUS_INET']
df.drop(uc, inplace=True, axis=1)
df

#Rename classes column
df.rename(columns={'FLAGGING': 'Class'},  inplace=True)

#Filling classes column with int value
df['Class'] = df['Class'].fillna(0)
df['Class'] = df['Class'].replace(['fraud'], 1)

#Summarizing fradulent cases
cases = len(df)
nonfraud_count = len(df[df.Class == 0])
fraud_count = len(df[df.Class == 1])
fraud_percentage = round(fraud_count/nonfraud_count*100, 2)

print('CASE COUNT')
print('----------------------------------------------------------')
print('Total number of cases are {}'.format(cases))
print('Number of Non-fraud cases are {}'.format(nonfraud_count))
print('Number of Non-fraud cases are {}'.format(fraud_count))
print('Percentage of fraud cases is {}'.format(fraud_percentage))
print('----------------------------------------------------------')

#Replace categorical column with int value
df['STO'].value_counts()
df['PRA_PASCA'].value_counts()
```

```
df['SPEED_INET'].value_counts()

# Replace categorical data
df['STO'] = df['STO'].replace(['PNK'], 1).replace(['BAL'], 2).replace(['MAT'], 3).r
eplace(['SUG'], 4).replace(['TMA'], 5).replace(['SUD'], 6).replace(['WTP'], 7).repl
ace(['ANT'], 8).replace(['MAR'], 9).replace(['PKN'], 10).replace(['TKA'], 11).repl
ace(['SIN'], 12).replace(['KIM'], 13).replace(['MAL'], 14).replace(['BLK'], 15).rep
lace(['BTN'], 16).replace(['SLY'], 17).replace(['JNP'], 18)
df['SPEED_INET'] = df['SPEED_INET'].replace(['200M'], 200).replace(['100M'],
100).replace(['50M'], 50).replace(['40M'], 40).replace(['20M'], 20).replace(['10M']
, 10)
df['PRA_PASCA'] = df['PRA_PASCA'].replace(['PRABAYAR'], 0).replace(['PA
SCABAYAR'], 1)
df


#Oversampling Data
ada = ADASYN(sampling_strategy=1.0, random_state=27)
x_train, y_train = ada.fit_resample(x_train, y_train)


#Min-Max Data Scalling
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)


#Data Split
x = df.drop('Class', axis = 1).values
y = df['Class'].values
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state
 = 0)
```

Lampiran 5 Menghitung waktu execute

```
from datetime import datetime
start = datetime.now()
end = datetime.now()
time_taken = end - start
print('Time: ',time_taken)
rf.get_params()
```

Lampiran 6 Algoritma Algoritma *Random Forest*

```
start = datetime.now()
#Random Forest
rf = RandomForestClassifier(n_estimators = 15, max_features = 1)
rf.fit(x_train, y_train)
rf_yhat = rf.predict(x_test)
end = datetime.now()
time_taken = end - start
print('Time: ',time_taken)
rf.get_params()
```

Lampiran 7 Algoritma XG Boost

```
start = datetime.now()

#XGBoost
reg = xgb.XGBClassifier(max_depth=3, n_estimators=250)
reg.fit(x_train, y_train)

y_xgb_pred = reg.predict(x_test)
print(y_test)
print(y_xgb_pred)

end = datetime.now()
time_taken = end - start
print('Time: ',time_taken)
reg.get_params()
```

Lampiran 8 Algoritma Logistic Regression

```
start = datetime.now()
#Logistic Regression
lr = LogisticRegression()
lr.fit(x_train, y_train)
lr_yhat = lr.predict(x_test)
end = datetime.now()
time_taken = end - start
print('Time: ',time_taken)
lr.get_params()
```

Lampiran 9 Algoritma Backpropagation Neural Network

```
start = datetime.now()
from sklearn.neural_network import MLPClassifier

#Data Scaling for BPNN
scaler = StandardScaler()
x_train_nn = scaler.fit_transform(x_train) #  transform
x_test_nn = scaler.transform(x_test) #  transform

#Multi Layer Perceptron Classifier
mlp = MLPClassifier(hidden_layer_sizes=125,
                    max_iter=480,
                    alpha=1e-4,
                    solver='sgd',
                    tol=1e-4,
                    random_state=1,
                    learning_rate='adaptive',
                    learning_rate_init=.1)


mlp.fit(x_train_nn, y_train)
mlp_yhat = mlp.predict(x_test_nn)
end = datetime.now()
time_taken = end - start
print('Time: ',time_taken)
mlp.get_params()
```

Lampiran 10 Confusion Matrix

```
import itertools # advanced tools
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

#Confusion Matrix
#Defining the plot function
def plot_confusion_matrix(cm, classes, title, normalize = False, cmap = plt.cm.Bl
ues):
    title = 'Confusion Matrix of {}'.format(title)
    if normalize:
        cm = cm.astype(float) / cm.sum(axis=1)[:, np.newaxis]

    plt.imshow(cm, interpolation = 'nearest', cmap = cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation = 45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment = 'center',
                 color = 'white' if cm[i, j] > thresh else 'black')

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

```
#Compute confusion matrix for the models
tree_matrix = confusion_matrix(y_test, tree_yhat, labels = [0, 1]) #Decision tree
rf_matrix = confusion_matrix(y_test, rf_yhat, labels = [0, 1]) #Random
Forest Tree
lr_matrix = confusion_matrix(y_test, lr_yhat, labels = [0, 1]) #Logistic Regression
mlp_matrix = confusion_matrix(y_test, mlp_yhat, labels = [0, 1]) #BPNN
xgb_matrix = confusion_matrix(y_test, y_xgb_pred, labels = [0, 1]) #XGBoost

#Plot the confusion matrix
plt.rcParams['figure.figsize'] = (6, 6)

#Decision tree
tree_cm_plot = plot_confusion_matrix(tree_matrix,
                        classes = ['Negative(0)','Positive(1)'],
                        normalize = False, title = 'Decision tree')
plt.savefig('tree_cm_plot.png')
plt.show()

#Random Forest tree
rf_cm_plot = plot_confusion_matrix(rf_matrix,
                        classes = ['Negative(0)','Positive(1)'],
                        normalize = False, title = 'Random Forest Tree')
plt.savefig('rf_cm_plot.png')
plt.show()

# Logistic regression
lr_cm_plot = plot_confusion_matrix(lr_matrix,
                        classes = ['Negative(0)','Positive(1)'],
                        normalize = False, title = 'Logistic Regression')
plt.savefig('lr_cm_plot.png')
```

```
plt.show()

# BPNN
mlp_cm_plot = plot_confusion_matrix(mlp_matrix,
                        classes = ['Negative(0)','Positive(1)'],
                        normalize = False, title = 'Backpropagation Neural Network')
plt.savefig('lr_cm_plot.png')
plt.show()

#XGBoost
xgb_cm_plot = plot_confusion_matrix(xgb_matrix,
                        classes = ['Negative(0)','Positive(1)'],
                        normalize = False, title = 'XG Boost')
plt.savefig('xgb_cm_plot.png')
plt.show()
```

Lampiran 11 Scoring Algoritma

```
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score

#Scoring
print('Precision Decision tree {}'.format(precision_score(y_test, tree_yhat)))
print('Recall Decision tree {}'.format(recall_score(y_test, tree_yhat)))
print('F1 Score Decision tree {}'.format(f1_score(y_test, tree_yhat)))
print('Accuracy Score Decision
tree {}'.format(accuracy_score(y_test, tree_yhat)))
print('_____
_____')
print('Precision Random Forest {}'.format(precision_score(y_test, rf_yhat)))
print('Recall Random Forest {}'.format(recall_score(y_test, rf_yhat)))
print('F1 Score Random Forest {}'.format(f1_score(y_test, rf_yhat)))
print('Accuracy Score Random
Forest {}'.format(accuracy_score(y_test, rf_yhat)))
print('_____
_____')
print('Precision Logistic Regression {}'.format(precision_score(y_test, lr_yhat)))
print('Recall Logistic Regression {}'.format(recall_score(y_test, lr_yhat)))
print('F1 Score Logistic Regression {}'.format(f1_score(y_test, lr_yhat)))
print('Accuracy Score Logistic Regression {}'.format(accuracy_score(y_test, lr_y
hat)))
print('_____
_____')
print('Precision Backpropagation Neural Network {}'.format(precision_score(y_te
st, mlp_yhat)))
```

```
print('Recall Backpropagation Neural Network {}'.format(recall_score(y_test, mlp
_yhat)))
print('F1 Score Backpropagation Neural Network {}'.format(f1_score(y_test, mlp
_yhat)))
print('Accuracy Score Backpropagation Neural Network {}'.format(accuracy_scor
e(y_test, mlp_yhat)))
print('_____
_____')
print('Precision XGBoost{}'.format(precision_score(y_test, y_xgb_pred)))
print('Recall XGBoost {}'.format(recall_score(y_test, y_xgb_pred)))
print('F1 Score XGBoost {}'.format(f1_score(y_test, y_xgb_pred)))
print('Accuracy Score XGBoost {}'.format(accuracy_score(y_test, y_xgb_pred)))
```

Lampiran 12 ROC Curve

```
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

#Defining ROC AUC
def plot_roc_curve(fpr, tpr):
    plt.plot(fpr, tpr, color='orange', label='ROC')
    plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend()
    plt.show()

#Defining probs of each model
#XG Boost
tree_model_prob = tree_model.predict_proba(x_test)
tree_model_prob = tree_model_prob[:, 1]
auc_tree_model = roc_auc_score(y_test, tree_model_prob)
print('AUC: %.2f' % auc_tree_model)
fpr_tree_model, tpr_tree_model, thresholds_tree_model = roc_curve(y_test, tree_
model_prob)
#Random Forest
rf_prob = rf.predict_proba(x_test)
rf_prob = rf_prob[:, 1]
auc_rf = roc_auc_score(y_test, rf_prob)
print('AUC: %.2f' % auc_rf)
fpr_rf, tpr_rf, thresholds_rf = roc_curve(y_test, rf_prob)
#Logistic Regression
lr_prob = lr.predict_proba(x_test)
```

```
lr_prob = lr_prob[:, 1]
auc_lr = roc_auc_score(y_test, lr_prob)
print('AUC: %.2f' % auc_lr)
fpr_lr, tpr_lr, thresholds_lr = roc_curve(y_test, lr_prob)
#BPNN
mlp_prob = mlp.predict_proba(x_test_nn)
mlp_prob = mlp_prob[:, 1]
auc_mlp = roc_auc_score(y_test, mlp_prob)
print('AUC: %.2f' % auc_mlp)
fpr_mlp, tpr_mlp, thresholds_mlp = roc_curve(y_test, mlp_prob)


#Plotting roc
#XG Boost
plot_roc_curve(fpr_tree_model, tpr_tree_model)
#Random Forest
plot_roc_curve(fpr_rf, tpr_rf)
#Logistic Regression
plot_roc_curve(fpr_lr, tpr_lr)
#BPNN
plot_roc_curve(fpr_mlp, tpr_mlp)
```

Lampiran 13 Data Generik

Sebagai contoh, dibuat data generik yang berisikan 4 variabel yang mewakili *internet speed, billing, usage,* dan *fraud* dengan masing-masing 10 buah data sebagai berikut:

No	Internet_Speed	Billing	Usage	Fraud
1	10	100	500	0
2	20	200	700	0
3	30	150	600	0
4	40	170	800	1
5	50	300	900	1
6	60	250	1000	1
7	70	280	1100	1
8	80	350	1200	1
9	90	400	1300	0
10	100	450	1400	0

Lampiran 14 Contoh Sederhana Cara Kerja *Random Forest* Menghasilkan Prediksi

Berikut adalah ilustrasi cara kerja *Random Forest* menggunakan data generik yang terlampir pada lampiran 13 diatas.

1. Billing<=100:

Subset 1 (Billing<=100): - Fraud=0: 1 – Fraud=1: 0 Gini Impurity = 1 - [(1)^2 + (0)^2] = 0

Subset 2 (Billing>100): - Fraud=0: 3 – Fraud=1: 6 Gini Impurity = 1 - [(3/9)^2 + (6/9)^2] = 0.44

Weighted Gini Impurity = (1/10)*0 + (9/10)*0.44 = 0.40

2. Billing<=150:

Sama seperti perhitungan sebelumnya, Weighted Gini Impurity = 0.42

3. Billing<=170:

Subset 1 (Billing<=170): - Fraud=0: 3 – Fraud=1: 1 Gini Impurity = 1 - [(3/4)^2 + (1/4)^2] = 0.38

Subset 2 (Billing>170): - Fraud=0: 1 – Fraud=1: 5 Gini Impurity = 1 - [(1/6)^2 + (5/6)^2] = 0.28

Weighted Gini Impurity = (4/10)*0.38 + (6/10)*0.28 = 0.32

Dan seterusnya, dilakukan perhitungan ini untuk semua nilai unik Billing. Dipilih nilai threshold mana yang memberikan Weighted Gini Impurity terendah sebagai threshold optimal untuk membagi data berdasarkan fitur Billing. Proses mencari nilai impurity ini biasanya dilakukan secara otomatis oleh algoritma Random Forest.Misalkan sudah ditentukan Billing memiliki *threshold* 200 memberikan penurunan *gini impurity* terbesar, maka:

1. Pembagian pertama: data dibagi menjadi dua grup berdasarkan internet_speed, grup pertama yang memiliki Billing <=200 dan grup lainnya Billing >200.
2. Pembagian selanjutnya: Mengulang proses pencarian *threshold* terbaik untuk setiap grup, tetapi dengan mempertimbangkan internet_speed dan Usage. Misalkan ditemukan pembagian Billing<=200 berdasarkan internet_speed <=20 dan membagi Billing>200 berdasarkan Usage<=800 memberikan penurunan impurity terbesar.
3. Proses lanjutan: proses dilanjutkan hingga mencapai kedalaman terdalam pohon atau tidak ada lagi pembagian yang bisa meningkatkan impurity.
4. Membuat prediksi: Setelah pohon keputusan selesai dibangun, pohon tersebut bisa digunakan untuk membuat prediksi. Untuk setiap sampel baru dimulai dengan pohon dan mengikuti cabang yang sesuai berdasarkan fitur sampel, prediksi untuk sampel tersebut adlah label mayoritas dari sampel training.

Berdasarkan pembahasan kita sebelumnya, kita memiliki data yang dibagi berdasarkan fitur Billing dengan threshold 200, dan kemudian dibagi lagi berdasarkan Usage dengan threshold 800 dan 1000.

1. Node pertama(root): Billing <=200?

    a. Jika ya, pergi ke node 2.
    b. Jika tidak, pergi ke node 3.

2. Node kedua: Usage <=800?

    a. Jika ya, pergi ke node 4.
    b. Jika tidak, pergi ke node 5.

3. Node ketiga: Usage<=1000?
    a. Jika ya, pergi ke node 6.
    b. Jika tidak, pergi ke node 7.

Setiap node daun model akan membuat prediksi berdasarkan mayoritas label kelas di data yang tersisa. Misalnya jika sebagian besar data di node 4 terdeteksi

sebagai fraud = 0, maka model akan memprediksi fraud=0 untuk data baru yang mencapai node ini.

Lampiran 15 Contoh Sederhana Cara Kerja XGBoost Menghasilkan Prediksi

Berikut adalah ilustrasi cara kerja *xgboost* menggunakan data generik yang terlampir pada tabel 1.

1. Langkah pertama adalah lakukan inisialisasi untuk nilai prediksi awal, pada kasus perhitungan ini, probabilitas awal semua data adalah 0,5 yang merupakan default dari algoritma XGBoost.
2. Hitung gradient dan hessian, Pertama tentukan fungsi loss:

   Log-Loss = -0 * log(0,5) – (1-0) * log(1 - 0,5)
   = -0 – log(0,5)
   = 0,693147

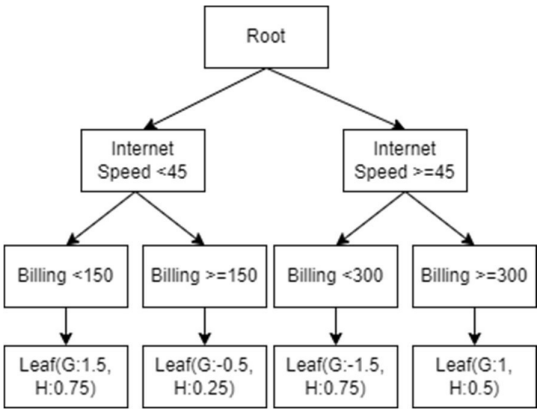
   Gradien adalah turunan pertama dari fungsi loss ini yang dihitung sebagai berikut:

   Gradien = y-p
   = 0 – 0,5
   = -0,5

   Hessian adalah turunan kedua dari fungsi loss ini, yang dihitung sebagai berikut:

   Hessian = p * (1-p)
   = 0,5 * (1-0,5)
   = 0,25

3. Tahap selanjutnya yaitu membentuk pohon baru berdasarkan nilai gradien dan hessian yang telah dihitung sebelumnya.



Selanjutnya, hitung nilai daun untuk setiap daun pada pohon menggunakan formula berikut :

Hitung bobot daun : $-\frac{\text{Gradient}}{\text{Hessian}+\lambda}$

Daun 1 = 1.5/(0.75+1) = 0,857

Daun 2 = -0.5/(0.25+1) = -0,4

Daun 3 = -1,5/(0,75+1) = - 0,857

Daun 4 = 1/(0,5+1) = 0,666

Nilai daun kemudian akan digunakan dalam langkah selanjutnya yaitu optimasi pohon dengan meminimalkan fungsi loss berikut setiap daunnya :

$$Obj = \sum_i [\, pred_i - y_i]^2 + \sum_j Reg_j$$

di mana $pred_i$ adalah prediksi untuk sampel i, y_i adalah label sebenarnya, dan Reg_j adalah regularisasi pada daun J

Regularisasi biasanya dihitung sebagai:

$$Reg = \gamma T + 1/2\, \lambda \sum_{j=1}^{T} w_j^2$$

Hitung Fungsi loss objektif :

= 2 x (0.857-0)^2 + 2 x $(-0.4 - 0)^2$ + 3 $(0{,}857 - 1)^2$ + 3$x(0{,}666 - 0)^2$ + 5,036227

= 2 x 0,734449 + 2 x 0,16 + 3 x 3,4489 + 3 x 0,443556 + 5,36227

= 1,468898 + 0,32 + 10,3467 + 1,330668 + 5,036227

= 18,502493

Fungsi loss objektif dari model ini adalah 18,502493

Ini adalah metrik yang akan kita gunakan untuk mengevaluasi dan membandingkan model ini dengan model lain dalam proses boosting.

4. Selanjutnya pembaruan prediksi, ini dilakukan dengan menambahkan prediksi dari pohon baru yang telah dibangun ke prediksi yang ada. Secara matematis ditulis sebagai berikut :
   Prediksi baru = prediksi lama + learning rate * prediksi pohon baru

Berdasarkan algoritma xgboost dengan 2 iterasi yang telah dibuat, menggunakan sampel berikut:

No	Internet_Speed	Billing	Usage	Fraud
1	47	250	800	?

Untuk data baru ini, jalur yang diambil adalah: Root -> [Billing >= 200] -> Leaf (0.666) untuk pohon pertama
Jalur yang diambil adalah: Root -> [Speed >= 30] -> Leaf (d) untuk pohon kedua
Asumsikan bahwa leaf(d) adalah 0.5 maka:

$$\text{Prediksi Akhir} = \eta \times (0.666 + 0.5) = 1 \times (0.666 + 0.5) = 1.166$$

Karena ini adalah tugas klasifikasi, akan digunakan fungsi sigmoid untuk mengubah prediksi ini menjadi probabilitas:

$$\text{Probabilitas} = \frac{1}{1+e^{-1.166}} \approx 0.76$$

Yang artinya data termasuk kedalam kelas 1 (Fraud)

Lampiran 16 Contoh Sederhana Cara Kerja Logistic Regression Menghasilkan Prediksi

Cara kerja *logistic regression* pada penelitian ini menggunakan metode *gradient descent.* Proses ini diawali dengan inisialisasi nilai bobot dan nilai bias serta penentuan *learning* rate.

Sebagai simulasi menggunakan data generik yang terlampir pada lampiran 13, berikut adalah cara kerja logistic regression dalam penelitian ini:

Pertama lakukan inisialisasi bobot dan bias masing b0 = -10, b1 = 0.2 (untuk internet_speed), b2 = 0.02 (untuk var 2), dan b3 = 0.01 (untuk Usage). Untuk baris pertama data, kita bisa menghitung probabilitas Fraud adalah 1 sebagai berikut:

$$P(Y=1) = 1 / (1 + e^{(-(b0 + b1*X1 + b2*X2 + b3*X3))})$$

Substitusi dengan nilai-nilai yang kita punya:

$$P(Y=1) = 1 / (1 + e^{-(-10 + 0.2*10 + 0.02*100 + 0.01*500)})$$

$$P(Y=1) = 1 / (1 + e^{-(-10 + 2 + 2 + 5)})$$

$$P(Y=1) = 1/(1 + e^{-1})$$

P = 0,731% (Terdeteksi fraud namun data yang terlampir tidak fraud, sehingga nilai bobot dan biasnya harus diubah)

Dalam contoh ini, dibuat asumsi sembarang mengenai nilai koefisien b0, b1, b2, dan b3 untuk demonstrasi. Sebenarnya, nilai-nilai ini harus ditemukan melalui proses pembelajaran dari data latih, ketika melatih model logistic regression dengan data yang benar-benar ada, nilai koefisien ditemukan dengan meminimalkan kesalahan antara prediksi model dan label yang sebenarnya dalam data latih.

Lampiran 17 Contoh Sederhana Cara Kerja BPNN Menghasilkan Prediksi

Menggunakan data generik yang terlampir pada lampiran 13, sebagai contoh diambil data pertama dengan

SPEED_INET = 10, BILLING = 100, USAGE = 500, IS_FRAUD = 0

Dengan menggunakan struktur jaringan sebagai berikut:

- Input layer : 3
- Hidden layer : 2
- Output layer : 1

Langkah 1: Inisialisasi bobot dan bias, sebagai contoh bobot dan bias diinisialisasikan dengan angka acak kecil berikut:

w1 = [0.1 , 0.2 , 0.3] , w2 =[0.4 , 0.5 , 0.6]

bias pada hidden layer :

b1= 0.1 , b2=0,2

bobot antara hidden dan output layer :

w3 = [0.7 , 0.8]

bias pada output layer:

b3 = 0.3

Langkah 2 : Feed forward

Misal diambil contoh 1 titik data : x = [10,100,500] dan y = 0

Dari Input ke Hidden Layer

$z1=(0.1\times10)+(0.2\times100)+(0.3\times500)+0.1$

$z1=1+20+150+0.1$

$z1=171.1$

$z2=(0.4\times10)+(0.5\times100)+(0.6\times500)+0.2$

$z2=4+50+300+0.2$

$z2=354.2$

$a1=\frac{1}{1+e^{-171.1}}$

$a2=\frac{1}{1+e^{-354.2}}$

Gunakan fungsi sigmoid:

h'1 = $\sigma(h1) = \frac{1}{1+e^{-171,1}} \approx 1$

h'2 = $\sigma(h2) = \frac{1}{1+e^{-35,2}} \approx 1$

Menghitung output neuron pada output layer:

$z_{output} = (0.7 \times a1) + (0.8 \times a2) + 0.3 \approx 1.8$

$a_{output} = \frac{1}{1+e^{-z_{output}}} = \frac{1}{1+e^{-1.8}} \approx 0.858$

Langkah 3 : Menghitung Kesalahan

Fungsi kerugian maka kesalahan pada output adalah :

$$\delta output = (y - a_{output}) \times a_{output} \times (1 - a_{output})$$

$$\delta output = (0 - 0.858) \times 0.858 \times (1 - 0.858)$$

$$\delta output \approx -0.1046$$

Nilai ini akan digunakan dalam langkah backpropagation untuk mengupdate bobot dan bias dari jaringan saraf tiruan

Hitung Kesalahan pada hidden layer,

Dimisalkan nilai $wo1 = 0.7$ dan nilai $wo2 = 0.8$

Maka kesalahan pada neuron 1 dan 2 pada hidden layer adalah:

$$\frac{\partial L}{\partial h1} = \frac{\partial L}{\partial h1} * wo1 * h'1 * (1 - h'1) = 0.858 * 0.7 * 1 * (1 - 1) = 0$$

$$\frac{\partial L}{\partial h2} = \frac{\partial L}{\partial h2} * wo2 * h'2 * (1 - h'2) = 0.858 * 0.8 * 1 * (1 - 1) = 0$$

Memperbarui bobot dan bias di seluruh jaringan

1. Update Bobot antara Hidden dan Output Layer

   Whidden-output baru = [0.7 , 0.8] + α × [− 0.1046] × [α1 , α2]

   Misalkan α=0.01, maka:

   Whidden-output baru = [0.7 − 0.001046 , 0.8 − 0.001046]

   Whidden-output baru =[0.698954 , 0.798954]

2. Update Bias pada Output Layer

$= 0.3 + \alpha\, x - 0.1046$

$= 0.2989540.298954$

3. Update Bobot antara Input dan Hidden Layer

= Tidak berubah dikarenakan nilai keduanya adalah 0

algoritma Backpropagation Neural Network (BPNN) atau biasanya disebut sebagai Backpropagation saja, bisa dikatakan algoritma untuk menyesuaikan bobot dan bias dalam jaringan neural

Langkah 4 Iterasi

Ulangi langkah 2 dan 3 sampai error mencapai batas yang diinginkan atau jumlah iterasi maksimum tercapai

Berdasarkan algoritma bpnn dengan menggunakan 1 titik data dengan 2 kali iterasi menggunakan data berikut :

No	Internet_Speed	Billing	Usage	Fraud
1	47	250	800	?

Didapatkan :

$a1 \approx 1$

$a2 \approx 1$

$z_{\text{output}} = 1.7968954$

$a_{\text{output}} \approx 0.858$ (Terdeteksi Fraud)

LEMBAR PERBAIKAN SKRIPSI

"IMPLEMENTASI DETEKSI FRAUD PENGGUNAAN JARINGAN INTERNET FIBER DI PERUSAHAAN PENYEDIA LAYANAN INTERNET"

OLEH:

ADY AHMADI SUWARDI
D121171517

Skripsi ini telah dipertahankan pada Ujian Akhir Sarjana tanggal 5 September 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. Amil Ahmad Ilham, S.T., M.IT.	
Sekretaris	A. Ais Prayogi Alimuddin, S.T., M.Eng	
Anggota	Prof. Dr. Eng. Intan Sari Areni, S.T., M.T.	
	Muhammad Alief Fahdal Imran Oemar, S.T., M.Sc	

Persetujuan Perbaikan oleh pembimbing:

Pembimbing	Nama	Tanda Tangan
I	Dr. Amil Ahmad Ilham, S.T., M.IT.	
II	A. Ais Prayogi Alimuddin, S.T., M.Eng	