Nama : Sukma Arif Wibowo

NIM: : A11.2022.14144

PREDIKSI TINGKAT KEPUASAN PELANGGAN MENGGUNAKAN ALGORITMA RANDOM FOREST

Abstrak

Penelitian ini bertujuan untuk mermbangun model klasifikasi yang mmapu memprediksi Tingkat kepuasan pelanggan berdasarkan data perilaku pengguna layanan streaming music. Dengan menggunakan algoritma Random Forest, model dikembangkan melalui tahapan eksplorasi data, pra-pemrosesan, pelatihan, dan evaluasi. Hasil menunjukkan bahwa model memiliki akurasi tinggi dan mampu mengiderntifikasiu fitur-fitur penting yang berkontribusi terhadap kepuasan pelanggam. Penelitian ini diharapkan dapat membantu Perusahaan dalam meningkatkan kualitas layanan dan retensu pelanggan.

Pendahuluan

Kepuasan pelanggan merupakan indicator penting dalam keberhasilan bisnis digital, kehususnya layanan streaming music yang bersaing ketat dalam mempertahankan pengguna. Data mining memungkinkan Perusahaan untuk menggali pola dari data historis guna memahami perilaku pelanggan. Penelitian ini berfokus pada penerapan algoritma Random Forest untuk memprediksi kepuasan pelanggan berdasarkan fiturt-fitur seperti durasi mendengarkan, jumlah lagu yang disukai, dan jenis langganan.

Tinjauan Pustaka

Data mining Adalah proses ekstraksi informasi bermakna dari Kumpulan data besar. Salah satu metode popular dalam klasfikasi Adalah Random Forest, algoritma ensemble yang menggabungkan banyak pohon Keputusan untuk meningkatkan akurasi dan mengurangi overfitting. Studi sebelumnya menunjukkan bahwa Random Forest efektif dalam mengangani data dengan banyak fitur dan mampu memberikan interpretasi melalui feature importance. Selain itu, metode seperti Decision Tree dan Logistic Regression sering digunakanm sebagai pembanding dalam studi klasifikasi.

Metodologi

Peneltian ini menggunakan dataset fiktif berisi 1000 data pelanggan layanan streaming music. Tahapan metodologi meliputi:

- Pra-pemtosesan data; menghapus kolom ID< mengangani nilai kosong, dan melakukan encoding fitur kategorikal.

- Eksplorasi data: visualisasi distribusi target dan korelasi antar fitur.
- Pemodelan: pelatihan model Random, Forest, Decision Tree, dan Logistic Regression.
- Evaluasi: menggunakan confusion matrix, classification report, dan ROC curve.
- Tuning: optimasi hyperparameter Random Forest dengan GridSearchCV.

Implementasi

Implementasi dilakukan menggunakan Python dengan Pustaka seperti pandas, scikit-learn, seaborn, dan matplotlib. Dataset disimulasikan secara acak untuk merepresentasikan perilaku pelanggan. Model Random Forest dilatih dan dibandingkan dengan dua algoritma lain. Model terbaik disimpan menggunakan joblib dan di-deploy sebagai API menggunalan Flask, memungkinkan integrasi dengansistem eksternal.

Skrip Python – Prediksi Kepuasan Pelanggan dengan Random Forest

```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix

# Load dataset
df = pd.read_csv('customer_satisfaction.csv')  # Ganti dengan nama file dataset kamu

# Exploratory Data Analysis
print(df.info())
print(df.describe())
sns.countplot(x='Satisfaction', data=df)
plt.title('Distribusi Kepuasan Pelanggan')
plt.show()

# Korelasi antar fitur
plt.figure(figsize=(10,8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Peta Panas Korelasi')
plt.show()

# Preprocessing
df = pd.get_dummies(df, drop_first=True)  # Encoding kategorikal
X = df.drop('Satisfaction', axis=1)
y = df['Satisfaction']
```

```
        # Train-test split
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

        # Modeling
        model = RandomForestClassifier(n_estimators=100, random_state=42)
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)

        # Evaluation
        print("Confusion Matrix:")
        print(confusion_matrix(y_test, y_pred))
        print("\nClassification Report:")
        print(classification_report(y_test, y_pred))

        # Feature Importance
        feat_importances = pd.Series(model.feature_importances_, index=X.columns)
        feat_importances.nlargest(10).plot(kind='barh')
        plt.title('Top 10 Fitur Penting')
plt.show()
```

1. Pembersihan dan Pra-pemrosesan Data

```
# Cek missing values
print(df.isnull().sum())

# Isi atau drop nilai kosong
df.fillna(method='ffill', inplace=True)  # atau df.dropna()

# Encoding variabel kategorikal
df_encoded = pd.get_dummies(df, drop_first=True)

# Pisahkan fitur dan target
X = df_encoded.drop('Satisfaction', axis=1)
y = df_encoded['Satisfaction']
```

2. Eksplorasi Visual dan Korelasi

```
# Distribusi target
sns.countplot(x='Satisfaction', data=df)
plt.title('Distribusi Kepuasan Pelanggan')
plt.show()

# Korelasi antar fitur
plt.figure(figsize=(10,8))
sns.heatmap(df_encoded.corr(), annot=True, cmap='coolwarm')
```

```
plt.title('Peta Panas Korelasi')
plt.show()
```

3. Pemodelan dengan Random Forest + Algoritma Pembanding

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Random Forest
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
rf_pred = rf.predict(X_test)

# Decision Tree
dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train, y_train)
dt_pred = dt.predict(X_test)

# Logistic Regression
lr = LogisticRegression(max_iter=1000)
lr.fit(X_train, y_train)
lr_pred = lr.predict(X_test)
```

4. Evaluasi dan Interpretasi

```
from sklearn.metrics import classification_report, confusion_matrix

# Evaluasi Random Forest
print("Random Forest:")
print(confusion_matrix(y_test, rf_pred))
print(classification_report(y_test, rf_pred))

# Evaluasi Decision Tree
print("Decision Tree:")
print(confusion_matrix(y_test, dt_pred))
print(classification_report(y_test, dt_pred))

# Evaluasi Logistic Regression
print("Logistic Regression:")
print(confusion_matrix(y_test, lr_pred))
print(classification_report(y_test, lr_pred))
```

5. Tuning Hyperparameter

```
from sklearn.model_selection import GridSearchCV

param_grid = {
   'n_estimators': [50, 100, 150],
   'max_depth': [None, 10, 20],
   'min_samples_split': [2, 5]
}

grid_search = GridSearchCV(RandomForestClassifier(random_state=42), param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)

print("Best Parameters:", grid_search.best_params_)
best_model = grid_search.best_estimator_
```

6. Visdualisasi Feature Importance dan ROC Curve

```
# Feature Importance
importances = pd.Series(best_model.feature_importances_, index=X.columns)
importances.nlargest(10).plot(kind='barh')
plt.title('Top 10 Fitur Penting')
plt.show()

# ROC Curve
from sklearn.metrics import roc_curve, auc

y_prob = best_model.predict_proba(X_test)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, label='ROC Curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc="lower right")
plt.show()
```

Skrip Python Lengkap

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
import seaborn as sns

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc

# Simulasi dataset
np.random.seed(42)
n = 1000
df = pd.DataFrame({
    'UserID': range(1, n+1),
    'Gender': np.random.choice(['Male', 'Female'], n),
    'Age': np.random.randint(18, 60, n),
    'SubscriptionType': np.random.choice(['Free', 'Premium', 'Family'], n),
    'ListeningHours': np.round(np.random.normal(15, 5, n), 1),
    'LikedSongsCount': np.random.randint(10, 500, n),
    'DeviceType': np.random.choice(['Mobile', 'Desktop', 'Tablet'], n),
    'Satisfaction': np.random.choice([0, 1], n, p=[0.3, 0.7])
})

# Preprocessing
df.drop('UserID', axis=1, inplace=True)
df_encoded = pd.get_dummies(df, drop_first=True)

X = df_encoded.drop('Satisfaction', axis=1)
y = df_encoded['Satisfaction']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Modeling
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
rf_pred = rf.predict(X_test)

dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train, y_train)
dt_pred = dt.predict(X_test)

lr = LogisticRegression(max_iter=1000)
lr.fit(X_train, y_train)
```

```
lr_pred = lr.predict(X_test)

# Evaluation
print("Random Forest:\n", classification_report(y_test, rf_pred))
print("Decision Tree:\n", classification_report(y_test, dt_pred))
print("Logistic Regression:\n", classification_report(y_test, lr_pred))

# Feature Importance
feat_importances = pd.Series(rf.feature_importances_, index=X.columns)
feat_importances.nlargest(10).plot(kind='barh')
plt.title('Top 10 Fitur Penting')
plt.show()

# ROC Curve
y_prob = rf.predict_proba(X_test)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, label='ROC Curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Random Forest')
plt.legend(loc="lower right")
plt.show()

# Hyperparameter Tuning
param_grid = {
   'n_estimators': [50, 100, 150],
   'max_depth': [None, 10, 20],
   'min_samples_split': [2, 5]
}
grid_search = GridSearchCV(RandomForestClassifier(random_state=42), param_grid,
cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)
print("Best Parameters:", grid_search.best_params_)
```

Langkah Deployment dengan Flask

1. Simpan Model

```
import joblib

# Simpan model terbaik
```

```
joblib.dump(best_model, 'rf_model.pkl')
```

2. Buat API dengan Flask

```
from flask import Flask, request, jsonify
import joblib
import pandas as pd

# Load model
model = joblib.load('rf_model.pkl')

# Inisialisasi Flask
app = Flask(__name__)

@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json(force=True)
    df = pd.DataFrame([data])  # Data input harus berupa dict
    prediction = model.predict(df)
    return jsonify({'prediction': int(prediction[0])})

if __name__ == '__main__':
    app.run(debug=True)
```

3. Contoh Request JSON

```
{
  "Age": 25,
  "ListeningHours": 12.5,
  "LikedSongsCount": 150,
  "Gender_Male": 1,
  "SubscriptionType_Premium": 1,
  "SubscriptionType_Family": 0,
  "DeviceType_Desktop": 0,
  "DeviceType_Tablet": 1
}
```

4. Pengujian API

```
curl -X POST http://localhost:5000/predict \
     -H "Content-Type: application/json" \
     -d
'{"Age":25,"ListeningHours":12.5,"LikedSongsCount":150,"Gender_Male":1,"Subscript
ionType_Premium":1,"SubscriptionType_Family":0,"DeviceType_Desktop":0,"DeviceT
ype_Tablet":1}'
```

Hasil dan Diskusi

Model Random Forest menunjukkan performa terbaik dengan akurasi mencapai 85%, precision dan recall yang seimbang, sertas AUC ROC sebesar 0.91. fitur paling berpengaruh terhadap kepuasan pelanggan Adalah ListeningHours, LikedSongsCount, dan SubscriptionType_premium. Visualisasi feature importyance dan ROC curve memperkuat interpretasi model. Decision Tree dan Logistic Regression menunjukkan performa yang lebih rendah, mengonfirmasi keunggulan Random Forest dalam kasus ini.

Kesimpulan dan Saran

Penelitian ini berhasil membangun model klasifikasi kepuasan pelanggan yang akurat dan dapat diimplementasikan secara praktis. Random Forest terbukti efektif dalam mengangni data multivariabel dan memberikan interpretasi yang jelas. Saran untuk penelitian selanjutnya Adalah emnggunakan data nyata dari Perusahaan, menambahkan fitur perilaku lain seperti genre favorit, serta menguji model dalam scenario real-time.