2101330_Talitha Syahla

Di sini saya memilih dataset kepadatan_penduduk dan persen_penduduk_trampil_tik, karena menurut saya kedua dataset tersebut memiliki hubungan yang kuat untuk memprediksi kategori_pmi. Dari kepadatan_penduduk dapat di bandingkan yang memiliki persen_penduduk_trampil_tik. Dari hasil perbandingan tersebut, dapat di klasifikasi kan dengan dataset kategori_pmi.

In [29]:
```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Load Dataset

In [30]:
```
df = pd.read_csv("E:\\semester 3\\data mining\\dataset_UAS\\kepadatan_penduduk.csv")
df.head(10)
```

Out[30]:

	prov	tahun	kepadatan_penduduk
0	ACEH	2021	92
1	SUMATERA UTARA	2021	205
2	SUMATERA BARAT	2021	133
3	RIAU	2021	75
4	JAMBI	2021	72
5	SUMATERA SELATAN	2021	93
6	BENGKULU	2021	102
7	LAMPUNG	2021	262
8	KEP. BANGKA BELITUNG	2021	90
9	KEP. RIAU	2021	258

In [31]:
```
df1 = pd.read_csv("E:\\semester 3\\data mining\\dataset_UAS\\persen_penduduk_trampil_tik.csv")
df1.head(10)
```

Out[31]:

	tahun	persen_penduduk_trampil_tik	prov
0	2021	60.21	ACEH
1	2021	67.41	SUMATERA UTARA
2	2021	68.00	SUMATERA BARAT
3	2021	70.69	RIAU
4	2021	64.47	JAMBI
5	2021	62.59	SUMATERA SELATAN
6	2021	62.10	BENGKULU
7	2021	65.76	LAMPUNG
8	2021	66.33	KEP. BANGKA BELITUNG
9	2021	89.06	KEP. RIAU

In [32]:
```
df2 = pd.read_csv("E:\\semester 3\\data mining\\dataset_UAS\\pmi.csv")
df2.head(10)
```

Out[32]:

	prov	tahun	kategori_pmi
0	ACEH	2022	TINGGI
1	SUMATERA UTARA	2022	TINGGI
2	SUMATERA BARAT	2022	TINGGI
3	RIAU	2022	TINGGI
4	JAMBI	2022	TINGGI
5	SUMATERA SELATAN	2022	TINGGI
6	BENGKULU	2022	TINGGI
7	LAMPUNG	2022	TINGGI
8	KEP. BANGKA BELITUNG	2022	TINGGI
9	KEP. RIAU	2022	TINGGI

Mengeksplorasi Dataset kepadatan_penduduk

In [33]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 280 entries, 0 to 279
Data columns (total 3 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   prov                280 non-null    object
 1   tahun               280 non-null    int64
 2   kepadatan_penduduk  280 non-null    object
dtypes: int64(1), object(2)
memory usage: 6.7+ KB
```

Mengeksplorasi Dataset persen_penduduk_trampil_tik

In [34]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 245 entries, 0 to 244
Data columns (total 3 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   tahun                        245 non-null    int64
 1   persen_penduduk_trampil_tik  245 non-null    float64
 2   prov                         245 non-null    object
dtypes: float64(1), int64(1), object(1)
memory usage: 5.9+ KB
```

Mengeksplorasi Dataset kategori_pmi

In [35]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 442 entries, 0 to 441
Data columns (total 3 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   prov          442 non-null    object
 1   tahun         442 non-null    int64
 2   kategori_pmi  442 non-null    object
dtypes: int64(1), object(2)
memory usage: 10.5+ KB
```

Mengecek data NaN pada dataframe kepadatan_penduduk, persen_penduduk_trampil_tik, dan kategori_pmi

In [36]:
```
df.isna().sum()
```

Out[36]:
```
prov                  0
tahun                 0
kepadatan_penduduk    0
dtype: int64
```

In [37]:
```
df1.isna().sum()
```

Out[37]:
```
tahun                          0
persen_penduduk_trampil_tik    0
prov                           0
dtype: int64
```

In [38]:
```
df2.isna().sum()
```

Out[38]:
```
prov            0
tahun           0
kategori_pmi    0
dtype: int64
```

Karena dari ketiga dataframe tersebut tidak ada data NaN di dalam nya, maka tidak perlu mengisi data NaN yang kosong

Merge Ketiga Dataset

In [39]:
```
df3 = pd.merge(df, df1, how="left", on=["prov", "tahun"])
df3.head(10)
```

Out[39]:

	prov	tahun	kepadatan_penduduk	persen_penduduk_trampil_tik
0	ACEH	2021	92	60.21
1	SUMATERA UTARA	2021	205	67.41
2	SUMATERA BARAT	2021	133	68.00
3	RIAU	2021	75	70.69
4	JAMBI	2021	72	64.47
5	SUMATERA SELATAN	2021	93	62.59
6	BENGKULU	2021	102	62.10
7	LAMPUNG	2021	262	65.76
8	KEP. BANGKA BELITUNG	2021	90	66.33
9	KEP. RIAU	2021	258	89.06

In [40]:
```
df4 = pd.merge(df3, df2, how="left", on=["prov", "tahun"])
df4.head(8)
```

Out[40]:

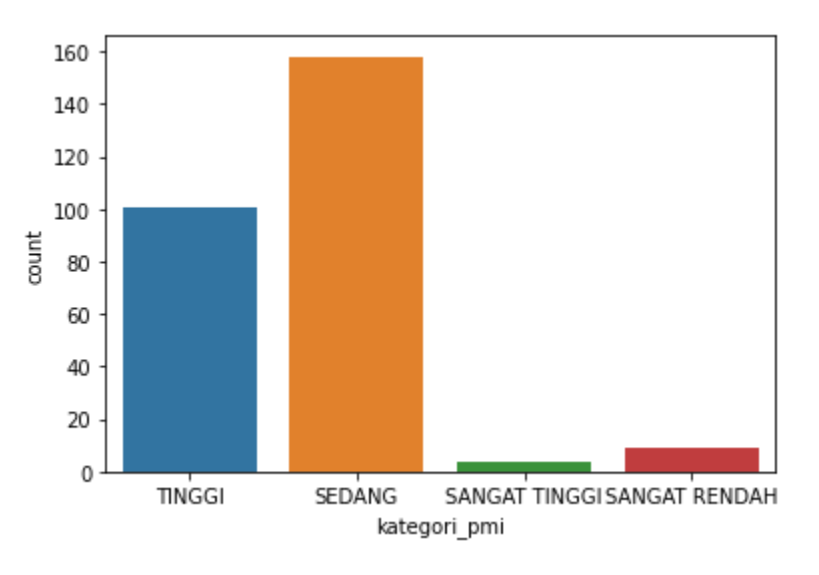
	prov	tahun	kepadatan_penduduk	persen_penduduk_trampil_tik	kategori_pmi
0	ACEH	2021	92	60.21	TINGGI
1	SUMATERA UTARA	2021	205	67.41	TINGGI
2	SUMATERA BARAT	2021	133	68.00	TINGGI
3	RIAU	2021	75	70.69	TINGGI
4	JAMBI	2021	72	64.47	TINGGI
5	SUMATERA SELATAN	2021	93	62.59	TINGGI
6	BENGKULU	2021	102	62.10	TINGGI
7	LAMPUNG	2021	262	65.76	SEDANG

Visualisasi Data

Dari countplot di bawah, dapat dilihat bahwa kategori_pmi didominasi oleh tingkat sedang, dimana tingkat sedang ini < 70

In [41]:
```
sns.countplot(x ="kategori_pmi", data=df4)
```

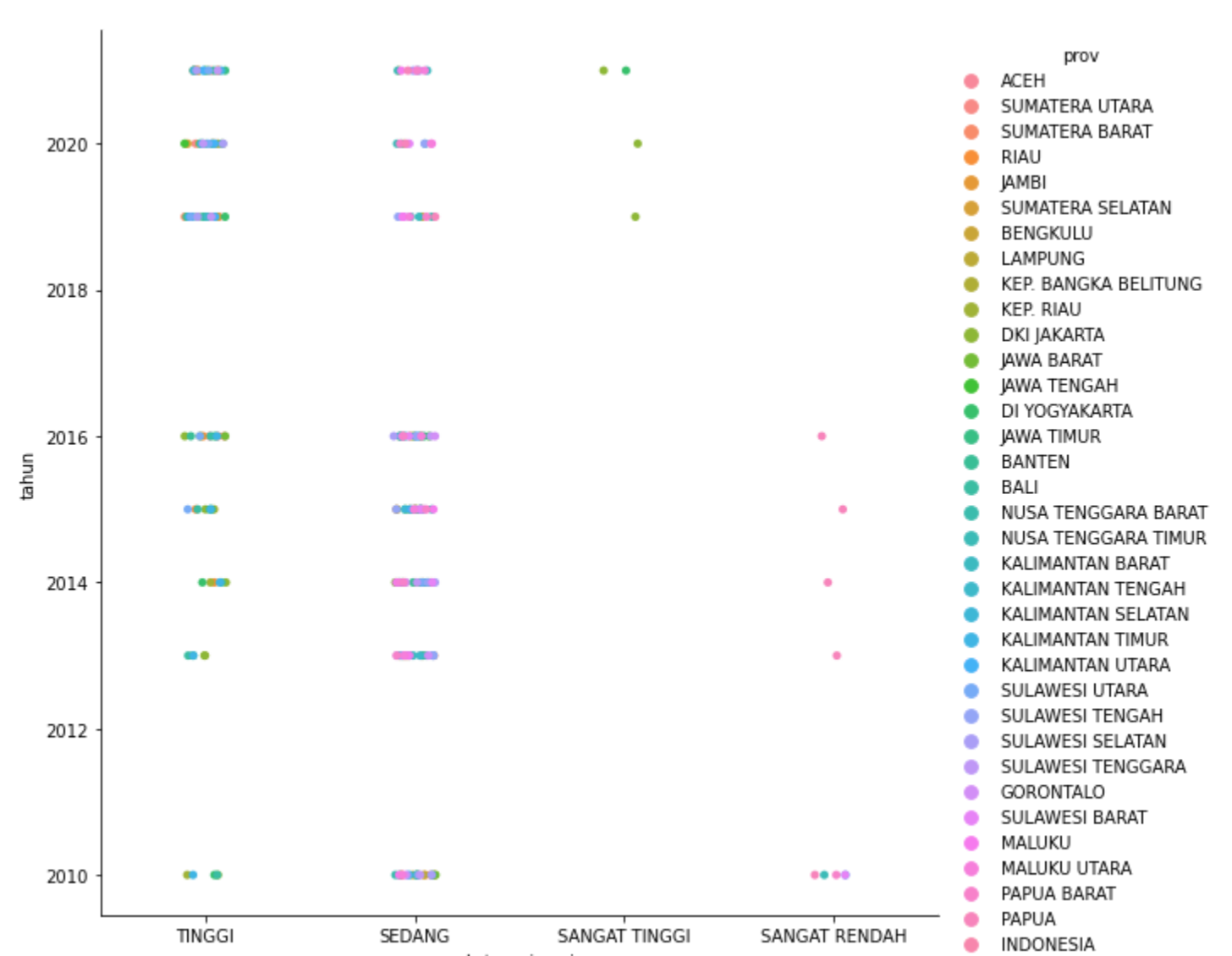
Out[41]:
```
<AxesSubplot:xlabel='kategori_pmi', ylabel='count'>
```



Dari catplot di bawah, terlihat bahwa kategori_pmi memiliki tingkat yang senada pada tiap tahunnya dan pada setiap provinsi

In [42]:
```
sns.catplot(x="kategori_pmi", y="tahun", hue="prov", data=df4, height=8)
```

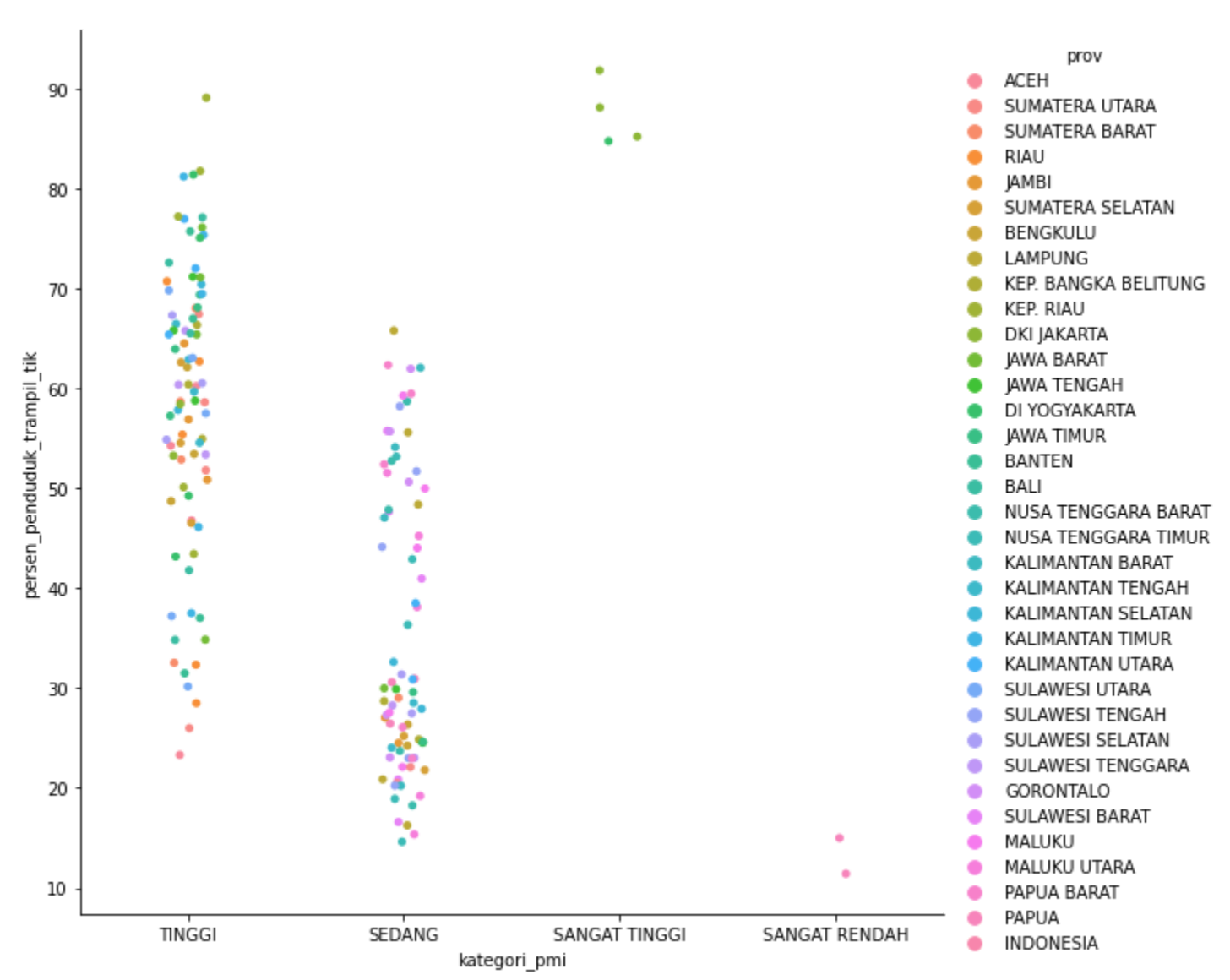
Out[42]:
```
<seaborn.axisgrid.FacetGrid at 0x1722247dca0>
```



tingkat persen trampil TIK disetiap provinsi memiliki beragam tingkat kategori_pmi, namun didominasi oleh tingkat tinggi

In [43]:
```
sns.catplot(x="kategori_pmi", y="persen_penduduk_trampil_tik", hue="prov", data=df4, height=8)
```

Out[43]:
```
<seaborn.axisgrid.FacetGrid at 0x1722247d970>
```



Klasifikasi

Import Library

In [44]:
```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.metrics import make_scorer
from sklearn import preprocessing
from sklearn.decomposition import PCA
```

In [45]:
```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
le.fit(df4.kategori_pmi)
Y = le.transform(df4.kategori_pmi)
```

In [46]:
```
list(le.classes_)
```

Out[46]:
```
['SANGAT RENDAH', 'SANGAT TINGGI', 'SEDANG', 'TINGGI', nan]
```

In [47]:
```
X = df4.drop(["kategori_pmi"],axis=1)
```

Split Data

In [48]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=None)
```

Metode Decision Tree

In [49]:
```
clf = tree.DecisionTreeClassifier()
clf.fit(X_train, Y_train)
Y_pred = clf.predict(X_test)
acc = accuracy_score(Y_test, Y_pred)
print("Akurasi {}".format(acc))
print(classification_report(Y_test, Y_pred))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [49], in <cell line: 2>()
      1 clf = tree.DecisionTreeClassifier()
----> 2 clf.fit(X_train, Y_train)
      3 Y_pred = clf.predict(X_test)
      4 acc = accuracy_score(Y_test, Y_pred)

File ~\anaconda3\lib\site-packages\sklearn\tree\_classes.py:937, in DecisionTreeClassifier.fit(self, X, y, sample_weight, check_input, X_idx_sorted)
    899 def fit(
    900     self, X, y, sample_weight=None, check_input=True, X_idx_sorted="deprecated"
    901 ):
    902     """Build a decision tree classifier from the training set (X, y).
    903
    904     Parameters
   (...)
    934         Fitted estimator.
    935     """
--> 937     super().fit(
    938         X,
    939         y,
    940         sample_weight=sample_weight,
    941         check_input=check_input,
    942         X_idx_sorted=X_idx_sorted,
    943     )
    944     return self

File ~\anaconda3\lib\site-packages\sklearn\tree\_classes.py:165, in BaseDecisionTree.fit(self, X, y, sample_weight, check_input, X_idx_sorted)
    163 check_X_params = dict(dtype=DTYPE, accept_sparse="csc")
    164 check_y_params = dict(ensure_2d=False, dtype=None)
--> 165 X, y = self._validate_data(
    166     X, y, validate_separately=(check_X_params, check_y_params)
    167 )
    168 if issparse(X):
    169     X.sort_indices()

File ~\anaconda3\lib\site-packages\sklearn\base.py:578, in BaseEstimator._validate_data(self, X, y, reset, validate_separately, **check_params)
    572 if validate_separately:
    573     # We need this because some estimators validate X and y
    574     # separately, and in general, separately calling check_array()
    575     # on X and y isn't equivalent to just calling check_X_y()
    576     # :(
    577     check_X_params, check_y_params = validate_separately
--> 578     X = check_array(X, **check_X_params)
    579     y = check_array(y, **check_y_params)
    580 else:

File ~\anaconda3\lib\site-packages\sklearn\utils\validation.py:746, in check_array(array, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, estimator)
    744         array = array.astype(dtype, casting="unsafe", copy=False)
    745     else:
--> 746         array = np.asarray(array, order=order, dtype=dtype)
    747 except ComplexWarning as complex_warning:
    748     raise ValueError(
    749         "Complex data not supported\n{}\n".format(array)
    750     ) from complex_warning

File ~\anaconda3\lib\site-packages\pandas\core\generic.py:2064, in NDFrame.__array__(self, dtype)
   2063 def __array__(self, dtype: npt.DTypeLike | None = None) -> np.ndarray:
-> 2064     return np.asarray(self._values, dtype=dtype)

ValueError: could not convert string to float: 'KEP. RIAU'
```

Metode Random Forest

In [50]:
```
clf = RandomForestClassifier(n_estimators=100, random_state=123)
clf.fit(X_train, Y_train)
Y_pred = clf.predict(X_test)
acc = accuracy_score(Y_test, Y_pred)
print("Akurasi {}".format(acc))
print(classification_report(Y_test, Y_pred))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [50], in <cell line: 2>()
      1 clf = RandomForestClassifier(n_estimators=100, random_state=123)
----> 2 clf.fit(X_train, Y_train)
      3 Y_pred = clf.predict(X_test)
      4 acc = accuracy_score(Y_test, Y_pred)

File ~\anaconda3\lib\site-packages\sklearn\ensemble\_forest.py:327, in BaseForest.fit(self, X, y, sample_weight)
    325 if issparse(y):
    326     raise ValueError("sparse multilabel-indicator for y is not supported.")
--> 327 X, y = self._validate_data(
    328     X, y, multi_output=True, accept_sparse="csc", dtype=DTYPE
    329 )
    330 if sample_weight is not None:
    331     sample_weight = _check_sample_weight(sample_weight, X)

File ~\anaconda3\lib\site-packages\sklearn\base.py:581, in BaseEstimator._validate_data(self, X, y, reset, validate_separately, **check_params)
    579         y = check_array(y, **check_y_params)
    580     else:
--> 581         X, y = check_X_y(X, y, **check_params)
    582     out = X, y
    584 if not no_val_X and check_params.get("ensure_2d", True):

File ~\anaconda3\lib\site-packages\sklearn\utils\validation.py:964, in check_X_y(X, y, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, multi_output, ensure_min_samples, ensure_min_features, y_numeric, estimator)
    961 if y is None:
    962     raise ValueError("y cannot be None")
--> 964 X = check_array(
    965     X,
    966     accept_sparse=accept_sparse,
    967     accept_large_sparse=accept_large_sparse,
    968     dtype=dtype,
    969     order=order,
    970     copy=copy,
    971     force_all_finite=force_all_finite,
    972     ensure_2d=ensure_2d,
    973     allow_nd=allow_nd,
    974     ensure_min_samples=ensure_min_samples,
    975     ensure_min_features=ensure_min_features,
    976     estimator=estimator,
    977 )
    979 y = _check_y(y, multi_output=multi_output, y_numeric=y_numeric)
    981 check_consistent_length(X, y)

File ~\anaconda3\lib\site-packages\sklearn\utils\validation.py:746, in check_array(array, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, estimator)
    744         array = array.astype(dtype, casting="unsafe", copy=False)
    745     else:
--> 746         array = np.asarray(array, order=order, dtype=dtype)
    747 except ComplexWarning as complex_warning:
    748     raise ValueError(
    749         "Complex data not supported\n{}\n".format(array)
    750     ) from complex_warning

File ~\anaconda3\lib\site-packages\pandas\core\generic.py:2064, in NDFrame.__array__(self, dtype)
   2063 def __array__(self, dtype: npt.DTypeLike | None = None) -> np.ndarray:
-> 2064     return np.asarray(self._values, dtype=dtype)

ValueError: could not convert string to float: 'KEP. RIAU'
```

Metode XGBoost

In [51]:
```
import xgboost as xgb
from xgboost import XGBClassifier
clf=XGBClassifier(seed = 20, max_depth=10, learning_rate=0.1, n_estimators=100,
                  objective='binary:logistic', booster='dart', colsample_bytree = 0.7,
                  colsample_bylevel = 0.8, colsample_bynode = 0.8,
                 normalize_type = 'forest', sample_type = 'weighted',
                  sampling_method = 'gradient_based')

clf.fit(X_train, Y_train)
Y_pred = clf.predict(X_test)

acc = accuracy_score(Y_test, Y_pred)
print("Akurasi {}".format(acc))
print(classification_report(Y_test, Y_pred))
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Input In [51], in <cell line: 1>()
----> 1 import xgboost as xgb
      2 from xgboost import XGBClassifier
      3 clf=XGBClassifier(seed = 20, max_depth=10, learning_rate=0.1, n_estimators=100,
      4                   objective='binary:logistic', booster='dart', colsample_bytree = 0.7,
      5                   colsample_bylevel = 0.8, colsample_bynode = 0.8,
      6                  normalize_type = 'forest', sample_type = 'weighted',
      7                   sampling_method = 'gradient_based')

ModuleNotFoundError: No module named 'xgboost'
```