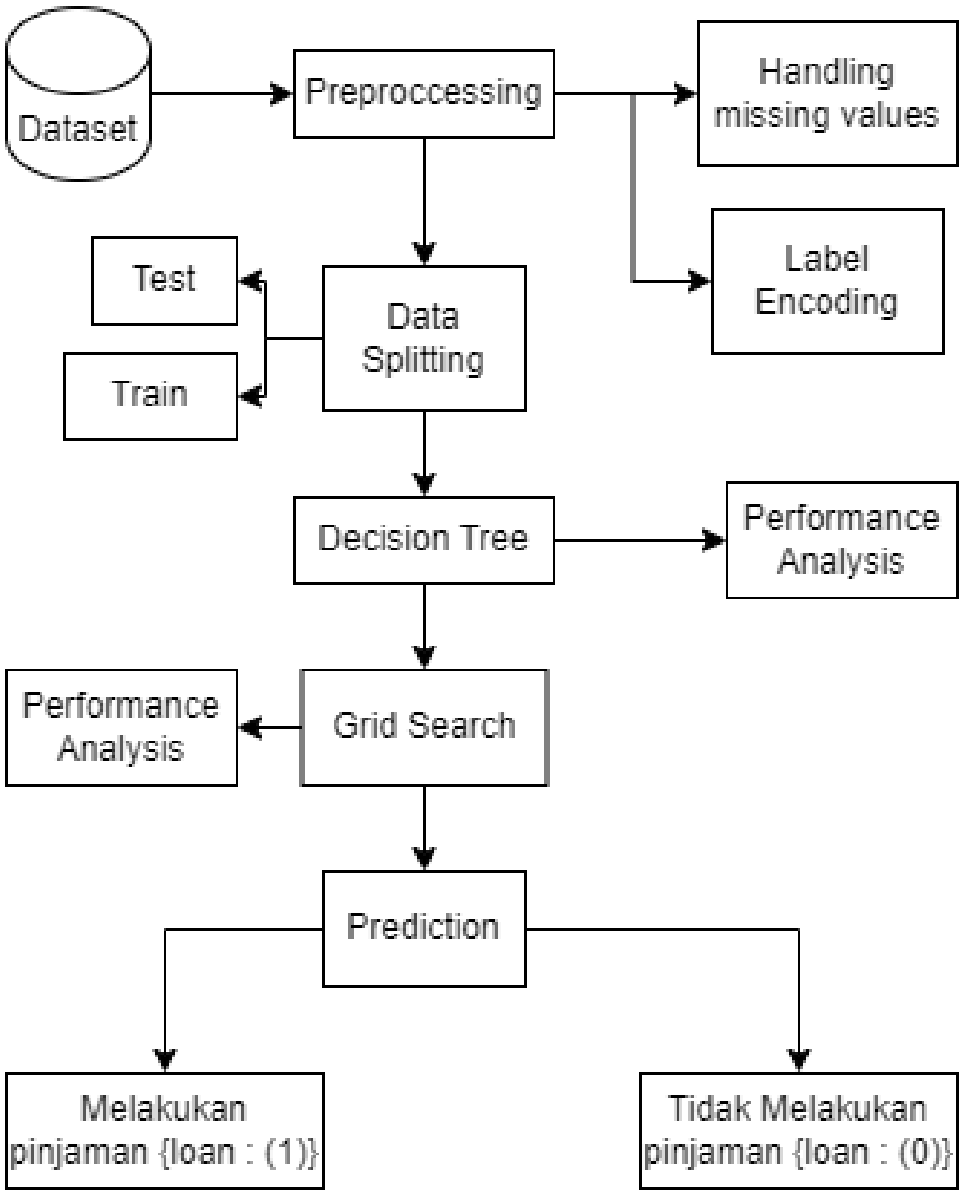
Rekrutment IT Support Bank BUMN

Rizki Rahman

Daftar Isi

Alur Pengerjaan



Alur pengerjaan dokumentasi code

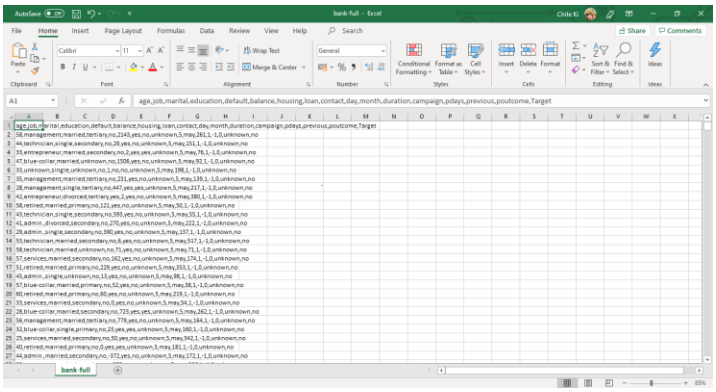
Sumber Data Kaggle

Kaggle yang saya ambil saya dapatkan dari open source Kaggle bernama bank-full.csv

Link Kaggle bank-full.csv : https://www.kaggle.com/datasets/krantiswalke/bankfullcsv

Informasi :

- Banyak data : 37902 data
- Atribut Data :
    - age (numerik)
    - job : jenis pekerjaan (kategorikal: admin.','kerah biru','pengusaha','pembantu rumah tangga','manajemen','pensiunan','wiraswasta','jasa','pelajar','teknisi','tidak bekerja','tidak diketahui')
    - marital : status perkawinan (kategorikal:'cerai','menikah','belum menikah','tidak diketahui'; catatan: 'cerai' berarti cerai hidup atau cerai mati)
    - education (kategorikal: 'dasar.4 tahun','dasar.6 tahun','dasar.9 tahun','SMA','tidak sekolah','tidak tamat SD','kursus profesional','sarjana','tidak diketahui')
    - default: apakah kredit Anda pernah macet? (kategorikal: 'tidak','ya','tidak diketahui')
    - balance: saldo tahunan rata-rata, dalam euro (numerik)
    - housing: memiliki kredit perumahan? (kategorikal: 'tidak','ya','tidak diketahui')
    - loan: memiliki pinjaman pribadi? (kategorikal: 'tidak','ya','tidak diketahui')
    - contact: jenis komunikasi yang digunakan (kategorikal:'seluler','telepon')
    - day: hari kontak terakhir dalam satu bulan (numerik 1 -31)
    - month: bulan kontak terakhir dalam satu tahun (kategorikal: 'jan', 'feb', 'mar', ..., 'nov', 'des')
    - duration: durasi kontak terakhir, dalam detik (numerik).
    - campaign: jumlah kontak yang dilakukan selama kampanye ini dan untuk klien ini (angka, termasuk kontak terakhir)
    - pdays: jumlah hari yang telah berlalu setelah klien terakhir kali dihubungi dari kampanye sebelumnya (angka; 999 berarti klien tidak dihubungi sebelumnya)
    - previous: jumlah kontak yang dilakukan sebelum kampanye ini dan untuk klien ini (angka)
    - poutcome: hasil dari kampanye pemasaran sebelumnya (kategorikal: 'gagal','tidak ada','sukses')
    - target: apakah klien telah berlangganan deposito berjangka? (biner: “ya”, “tidak”)



Data mentah pada excel

Dokumentasi Code

```python
import pandas as pd
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt

import warnings

from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn import metrics
from sklearn.metrics import precision_recall_curve

from sklearn import tree
from sklearn.model_selection import GridSearchCV

from sklearn.tree import DecisionTreeClassifier #untuk membangun model

from sklearn.model_selection import train_test_split #split data

warnings.filterwarnings('ignore') # To supress warnings
```

Import library yang akan digunakan, disini saya menggunakan pandas untuk penggunaan dataframe, numpy untuk kategori numerik, lalu visualiasi disini menggunakan seaborn dan matplotlib.pyplot. Untuk mengecek performa saya menggunakan sklearn.metrics dan algoritma decision tree serta Grid search untuk hyperparameter. Lalu split data menggunakan train_test_split.

```
[4] df = pd.read_csv('/content/bank-full.csv')
```

```
[5] df.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	Target
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261.0	1.0	-1.0	0.0	unknown	no
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151.0	1.0	-1.0	0.0	unknown	no
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76.0	1.0	-1.0	0.0	unknown	no
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92.0	1.0	-1.0	0.0	unknown	no
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198.0	1.0	-1.0	0.0	unknown	no

Import data yang diambil dari Kaggle lalu melihat isi data 5 pretama menggunakan head()

```
[9] df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 37902 entries, 0 to 37901
Data columns (total 17 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   age         37902 non-null  int64
 1   job         37902 non-null  object
 2   marital     37902 non-null  object
 3   education   37902 non-null  object
 4   credit      37902 non-null  object
 5   balance     37902 non-null  int64
 6   housing     37902 non-null  object
 7   loan        37902 non-null  object
 8   contact     37902 non-null  object
 9   day         37902 non-null  int64
 10  month       37902 non-null  object
 11  duration    37901 non-null  float64
 12  campaign    37901 non-null  float64
 13  pdays       37901 non-null  float64
 14  previous    37901 non-null  float64
 15  poutcome    37901 non-null  object
 16  subs_depo   37901 non-null  object
dtypes: float64(4), int64(3), object(10)
memory usage: 4.9+ MB
```

Cek info data apakah ada data yang null, pada case ini tiap column tidak memiliki data null

```
#memilih categorical feature
categorical=[]
for col, value in df.items():
  if value.dtype == 'object':
    categorical.append(col)

#store numerical kolum list
df2 = df[categorical]
```

```
[11] df2.head()
```

	job	marital	education	credit	housing	loan	contact	month	poutcome	subs_depo
0	management	married	tertiary	no	yes	no	unknown	may	unknown	no
1	technician	single	secondary	no	yes	no	unknown	may	unknown	no
2	entrepreneur	married	secondary	no	yes	yes	unknown	may	unknown	no
3	blue-collar	married	unknown	no	yes	no	unknown	may	unknown	no
4	unknown	single	unknown	no	no	no	unknown	may	unknown	no

Memilih data categorical berupa object untuk dipisahkan dan dianalisa

```
[13] #convert to numerical
     from sklearn import preprocessing
     le = preprocessing.LabelEncoder()

     df2_feature_select = df2.copy()

     for columnName in df2_feature_select:
       df2_feature_select[columnName] = le.fit_transform(df2_feature_select[columnName])

     df2_feature_select.head()
```

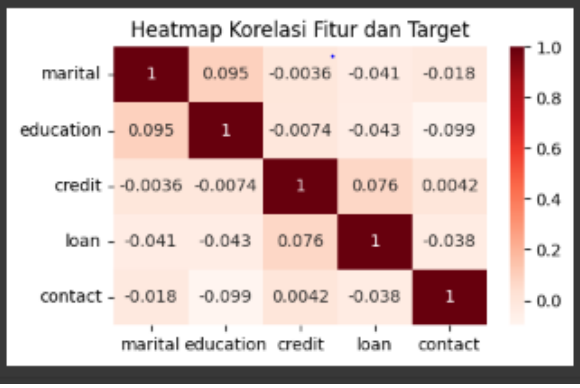
	job	marital	education	credit	housing	loan	contact	month	poutcome	subs_depo
0	4	1	2	0	1	0	2	9	3	0
1	9	2	1	0	1	0	2	9	3	0
2	2	1	1	0	1	1	2	9	3	0
3	1	1	3	0	1	0	2	9	3	0
4	11	2	3	0	0	0	2	9	3	0

Mengkonversi data kategorikal ke dalam data numerik

```
[14] # Membuat korelasi antara fitur dan target
     cor = df2_feature_select.corr()
     cor_target = abs(cor['loan'])
     relevant_features = cor_target[cor_target > 0.03]
     relevant_features
```

```
marital      0.041293
education    0.043469
credit       0.075604
loan         1.000000
contact      0.037670
Name: loan, dtype: float64
```

```
     # Memilih fitur yang relevan
     relevant_df = df2_feature_select[relevant_features.index]

     # Membuat heatmap
     plt.figure(figsize=(5, 3))
     sns.heatmap(relevant_df.corr(), annot=True, cmap=plt.cm.Reds)
     plt.title('Heatmap Korelasi Fitur dan Target')
     plt.show()
```



Membuat korelasi data untuk data yang saling berkaitan, korelasi yang digunakan sebesar 0,03

```
[40] relevant_features_col = ['job','marital','education','credit','housing', 'subs_depo','loan']
     selected_df= df2[relevant_features_col]

[17] #meghapus kolom yang tidak digunakan di model
     df3 = df2_feature_select.drop(columns=["contact", "month","poutcome"])

[18] X_dt = df3.drop('loan', axis=1)
     y_dt = df3['loan']
```

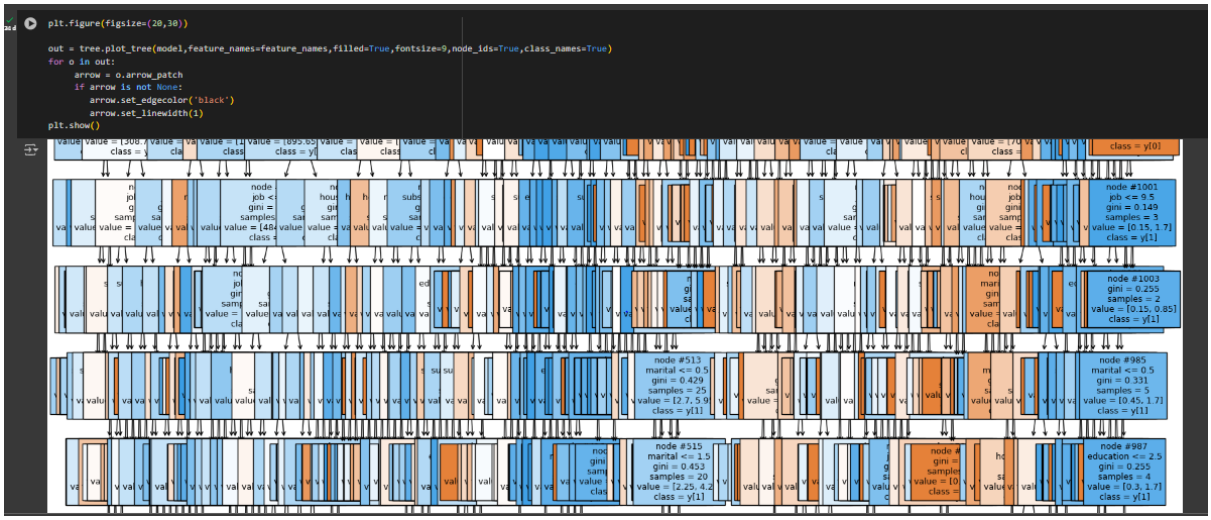
Memasukkan fitur relevant kolom lalu membuang data yang tidak digunakan. Setelah itu menentukan variable X dan y untuk dilakukan splitting data.

```
[19] oneHotCols=X_dt.select_dtypes(exclude='number').columns.to_list()
     X_dt=pd.get_dummies(X_dt,columns=oneHotCols,drop_first=True)
     # Spliting data set
     X_train_dt, X_test_dt, y_train_dt, y_test_dt = train_test_split(X_dt, y_dt, test_size=0.3, random_state=1, stratify=y_dt)

##  Function to calculate recall score
def get_recall_score(model):
    '''
    model : classifier to predict values of X

    '''
    ytrain_predict = model.predict(X_train_dt)
    ytest_predict = model.predict(X_test_dt)
    # accuracy on training set
    print("\x1b[0;30;47m \033[1mAccuracy : Train :\033[0m",
          model.score(X_train_dt,y_train_dt),
          "\x1b[0;30;47m \033[1mTest:\033[0m",
          model.score(X_test_dt,y_test_dt))
# accuracy on training set
    print("\x1b[0;30;47m \033[1mRecall   : Train :\033[0m",
          metrics.recall_score(y_train_dt,ytrain_predict),
          "\x1b[0;30;47m \033[1mTest:\033[0m",
          metrics.recall_score(y_test_dt,ytest_predict))
```

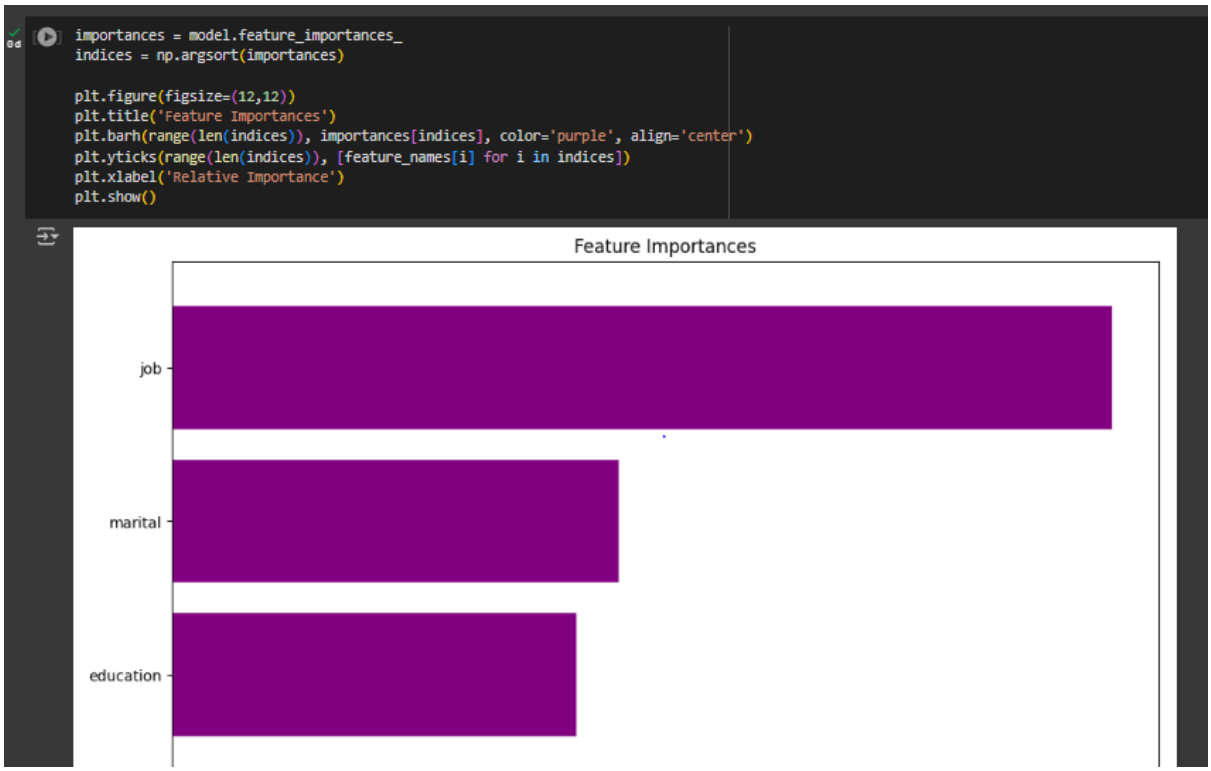
Menentukan data numerik dan memasukkannya ke dalam list outHotCols, setelah itu data tersebut dilakukan training testing, data training yang digunakan sebesar 70% dan data testing sebesar 30%. Lalu mendefinisikan fuction score dengan parameter model untuk melihat performa.

```
[22] #since data is imbalanced adding weights
     model = DecisionTreeClassifier(criterion = 'gini',class_weight={0:0.15,1:0.85}, random_state=1)
     model.fit(X_train_dt, y_train_dt)
     get_recall_score(model)

 Accuracy : Train : 0.4829821717990275 Test: 0.46249230498636884
 Recall   : Train : 0.7609578789822535 Test: 0.7206982543640897

[23] column_names = list(X_dt.columns)
     feature_names = column_names
     print(feature_names)

 ['job', 'marital', 'education', 'credit', 'housing', 'subs_depo']
```

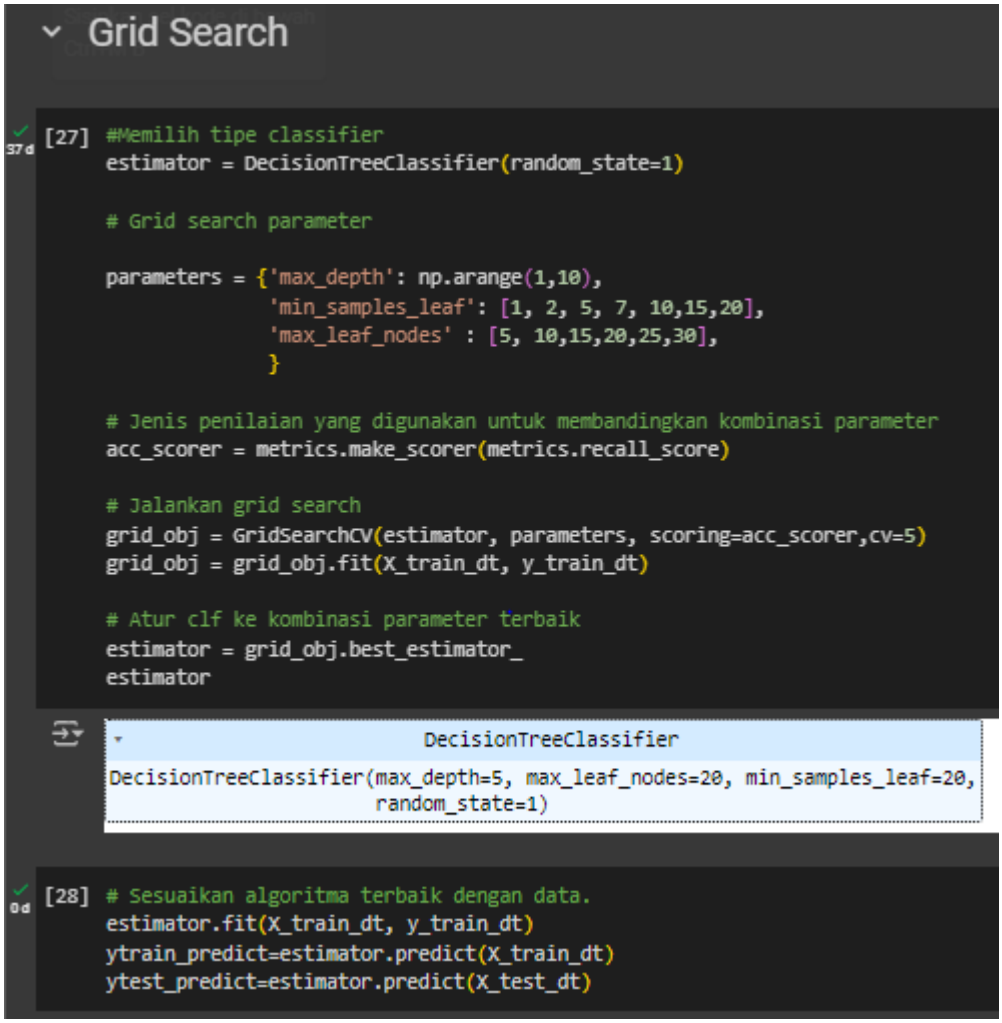
Didapat performa dan fitur yang digunakan



Didapatkan hasil visual dari decision tree namun cabang yang dihasilkan terlalu banyak, disini saya mencoba menerapkan grid search agar performa yang dihasilkan stabil

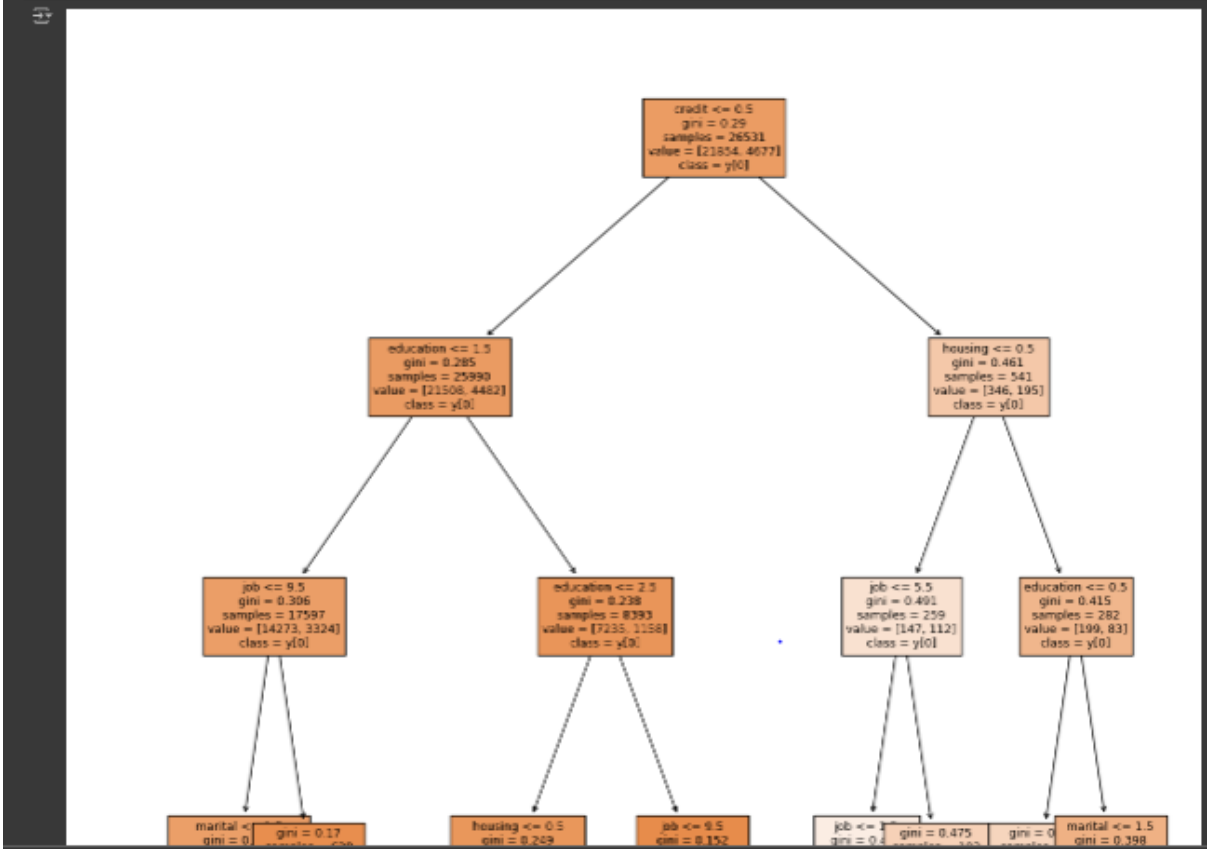


Hasil yang ditampilkan feature yang paling berpengaruh yaitu pada kolom job.



Hyperparameter yang digunakan berupa Grid Search dengan parameter sample untuk decision tree dan didapat parameter yang stabil digunakan pada decisiontreeclassifier.

```
[29] plt.figure(figsize=(15,20))

    out = tree.plot_tree(estimator,feature_names=feature_names,filled=True,fontsize=9,node_ids=False,class_names=True)
    for o in out:
        arrow = o.arrow_patch
        if arrow is not None:
            arrow.set_edgecolor("black")
            arrow.set_linewidth(1)
    plt.show()
```



Hasil yang didapatkan sedikit lebih membaik dan node yang dihasilkan tidak terlalu banyak.

```
[30] importances = estimator.feature_importances_
     indices = np.argsort(importances)

     plt.figure(figsize=(12,12))
     plt.title('Feature Importances')
     plt.barh(range(len(indices)), importances[indices], color='violet', align='center')
     plt.yticks(range(len(indices)), [feature_names[i] for i in indices])
     plt.xlabel('Relative Importance')
     plt.show()
```

Untuk feature yang berpengaruh sendiri berubah yaitu menjadi kolom education

```
[31] get_recall_score(estimator)

     Accuracy : Train : 0.8240548791979194 Test: 0.823234544015478
     Recall   : Train : 0.0074834295488561044 Test: 0.005486284289276808

[32] y_pred = estimator.predict(X_test_dt)
     print(classification_report(y_test_dt,y_pred))

                   precision    recall  f1-score   support

                0       0.82      1.00      0.90      9366
                1       0.41      0.01      0.01      2005

         accuracy                           0.82     11371
        macro avg       0.62      0.50      0.46     11371
     weighted avg       0.75      0.82      0.75     11371

[33] misclass_df = X_test_dt.copy()

[34] misclass_df['Actual']=y_test_dt
     misclass_df['Predicted'] = y_pred
```

Untuk performa yang dihasilkan sendiri lebih besar dari sebelumnya dan meningkat secara signifikan.

Insight Kesimpulan

Dari Analisa yang dilakukan didapat hasil untuk melihat kelayakan nasabah bank apakah layak untuk melakukan pengajuan pinjaman ke bank, berdasarkan factor sesuai data yang saya ambil dari Kaggle. Faktor tersebuh yaitu job, martial, education, credit, hosuing, dan subs_depo mereka. Dengan hasil akhir sebagai berikut :

```
Row :
 job          blue-collar
marital          married
education      secondary
credit                no
housing              yes
subs_depo             no
loan                  no
Name: 50, dtype: object
```

Row ini menunjukkan data indeks ke 50 dengan kriteria tersebut. Lalu dilakukan prediksi dan disamakan dengan actual data seperti berikut :

```
Actual :  0
Prediction :  0    0
Name: 50, dtype: int64
```

Hasil yang didapat dari prediksi 0 sama dengan actual nya yaitu 0 yang bearti nasabah tersebut tidak layak melakukan pinjaman ke bank.

Lalu apabila ingin melakukan prediksi secara langsung dapat menggunakan definisi function dengan kriteria sebelumnya, lalu menggunakan numerik list kriteria tersebut sebagai berikut

```
Row :
 job          blue-collar
marital          married
education      secondary
credit                no
housing              yes
subs_depo             no
loan                  no
Name: 50, dtype: object
```

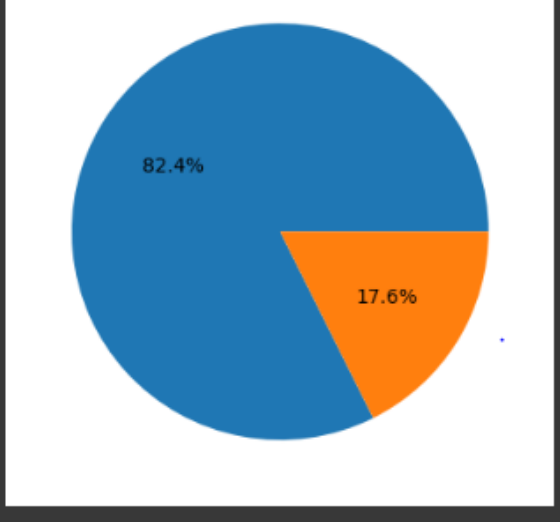
```
Row :
 job          1
marital       1
education     1
credit        0
housing       1
loan          0
subs_depo     0
Name: 50, dtype: int64
```

Dari kedua gambar itu berasal dari data sama yaitu indeks ke 50 hanya saja function yang dipakai menggunakan feature numerik. Untuk melakukan prediksi dapat menggunakan function sebagai berikut :

-
```
[46] def predict_one(job, marital, education, credit, housing, subs_depo):
         print('Prediction : ', estimator.predict([[job, marital, education, credit, housing, subs_depo]]))
```
- **predict_one(1, 1, 1, 0, 1, 0)**
  - parameter angka dari kiri ke kanan -> (job, martial, education, credit, hosuing, subs_depo)
-
```
Actual :  0
Prediction :  [0]
```
- Hasil yang didapat sama
  - Actual : 0
  - Prediction : 0

```
[35] plt.pie(data=misclass_df,x=misclass_df["Actual"].value_counts(),autopct='%1.1f%%')
```

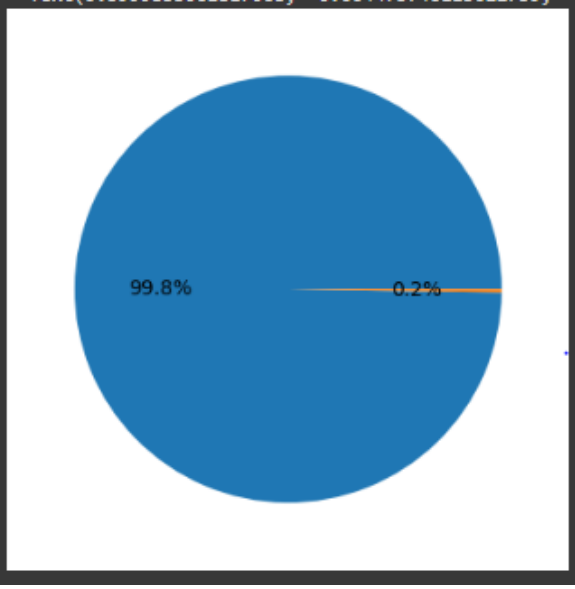
```
([<matplotlib.patches.Wedge at 0x79e34ee19c00>,
  <matplotlib.patches.Wedge at 0x79e34ee19a50>],
 [Text(-0.9355022659711025, 0.5786497302884818, ''),
  Text(0.9355022659711025, -0.5786497302884819, '')],
 [Text(-0.510273963256965, 0.31562712561189915, '82.4%'),
  Text(0.510273963256965, -0.3156271256118992, '17.6%')])
```



Performa dihasilkan dari actual sendiri 82.4%.

```
[36] plt.pie(data=misclass_df,x=misclass_df["Predicted"].value_counts(),autopct='%1.1f%%')
```

```
([<matplotlib.patches.Wedge at 0x79e34d5f86d0>,
  <matplotlib.patches.Wedge at 0x79e34eebe6e0>],
 [Text(-1.0999693948945386, 0.008205503966404142, ''),
  Text(1.0999693946664626, -0.008205534540541688, '')],
 [Text(-0.599983306306112, 0.00447572943622044, '99.8%'),
  Text(0.5999833061817068, -0.004475746113022739, '0.2%')])
```



Lalu untuk performa data yang diprediksi 99.8%

Decision Tree & Grid Search

Decision tree adalah model prediktif yang mengambil keputusan dengan membagi ruang fitur menjadi segmen-segmen yang lebih kecil. Proses ini melibatkan pemilihan fitur terbaik pada setiap langkah untuk membagi data menjadi dua bagian, sehingga memaksimalkan informasi yang diperoleh pada setiap langkah.

Alasannya sendiri menggunakan decision tree karena salah satu algoritma yang mudah diinterpretasi. Struktur pohon yang dihasilkan dapat dengan mudah dimengerti oleh manusia. Lalu memberikan informasi tentang pentingnya fitur dalam pembuatan keputusan. Ini dapat membantu bank untuk memahami faktor-faktor apa yang paling mempengaruhi keputusan pemberian pinjaman.

Grid Search berisi semua kombinasi hyperparameter yang mungkin untuk sebuah model. Misalnya, untuk decision tree, hyperparameter dapat termasuk kedalaman maksimum pohon, jumlah minimum sampel untuk pemisahan simpul, dll. Grid Search sendiri memilih kombinasi hyperparameter yang memberikan kinerja terbaik sesuai dengan metrik evaluasi yang dipilih.

Alasan untuk melakukan tuning menggunakan Grid search karena memungkinkan kita untuk mencari kombinasi hyperparameter yang optimal untuk model decision tree. Hal ini membantu meningkatkan kinerja model dengan menemukan hyperparameter terbaik untuk dataset yang diberikan. Lalu dapat menyesuaikan parameter model yang membantu menangani masalah ketidakseimbangan kelas dalam dataset.