BINAR

Data Science

Presentation

Latar Belakang

Persaingan meningkatkan internet service di perusahaan telekomunikasi merupakan dampak dari perilaku komunikasi masyarakat saat ini.

Memilih provider yang sesuai merupakan hak bagi pelanggan.

Peralihan ini dapat menyebabkan berkurangnya pendapatan bagi perusahaan telekomunikasi sehingga penting untuk ditangani.

TUJUAN

Membangun Model untuk melakukan klasifikasi pelanggan yang beralih atau tidak

Eksplorasi Data

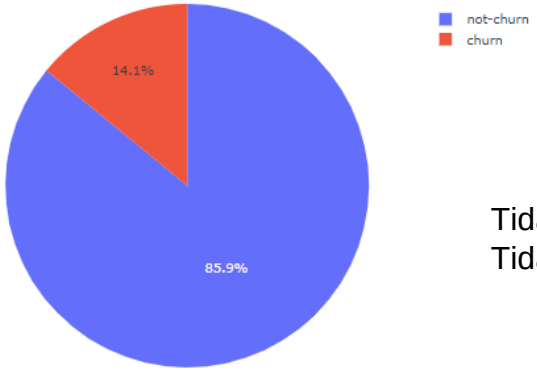
Eksplorasi Data

Setelah import data dilanjutkan eksplorasi

Jumlah data train **4250** dengan 20 kolom (19 fitur dan 1 label)

```
Index(['state', 'account_length', 'area_code', 'international_plan',
       'voice_mail_plan', 'number_vmail_messages', 'total_day_minutes',
       'total_day_calls', 'total_day_charge', 'total_eve_minutes',
       'total_eve_calls', 'total_eve_charge', 'total_night_minutes',
       'total_night_calls', 'total_night_charge', 'total_intl_minutes',
       'total_intl_calls', 'total_intl_charge',
       'number_customer_service_calls', 'churn'],
      dtype='object')
(4250, 20)
```

598 beralih dan 3652 tidak



Tidak ada data kosong
Tidak ada duplikasi

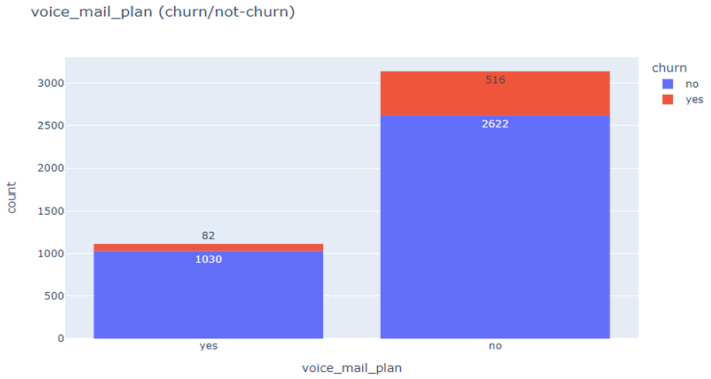
5 Categoric 15 numeric

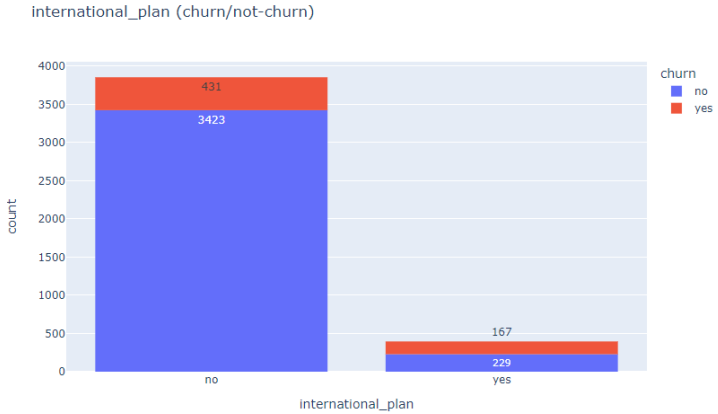
```
df_train.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4250 entries, 0 to 4249
Data columns (total 20 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   state                         4250 non-null   object
 1   account_length                4250 non-null   int64
 2   area_code                     4250 non-null   object
 3   international_plan            4250 non-null   object
 4   voice_mail_plan               4250 non-null   object
 5   number_vmail_messages         4250 non-null   int64
 6   total_day_minutes             4250 non-null   float64
 7   total_day_calls               4250 non-null   int64
 8   total_day_charge              4250 non-null   float64
 9   total_eve_minutes             4250 non-null   float64
 10  total_eve_calls               4250 non-null   int64
 11  total_eve_charge              4250 non-null   float64
 12  total_night_minutes           4250 non-null   float64
 13  total_night_calls             4250 non-null   int64
 14  total_night_charge            4250 non-null   float64
 15  total_intl_minutes            4250 non-null   float64
 16  total_intl_calls              4250 non-null   int64
 17  total_intl_charge             4250 non-null   float64
 18  number_customer_service_calls 4250 non-null   int64
 19  churn                         4250 non-null   object
dtypes: float64(8), int64(7), object(5)
memory usage: 664.2+ KB
```

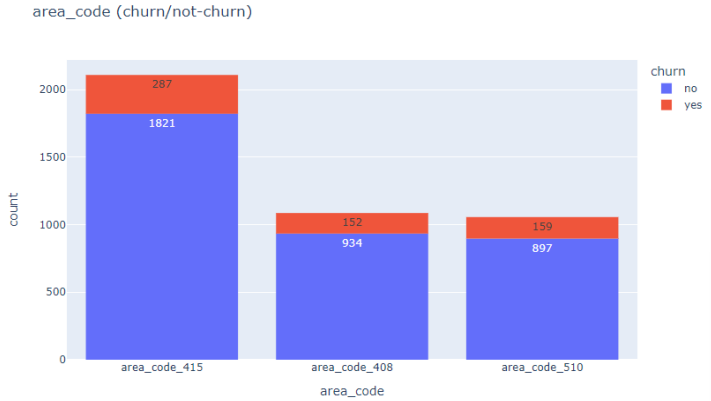
Eksplorasi Data

BINAR

```
international_plan     2
voice_mail_plan        2
churn                  2
area_code              3
state                 51
dtype: int64
```







Eksplorasi Data

waktu mempengaruhi churn?

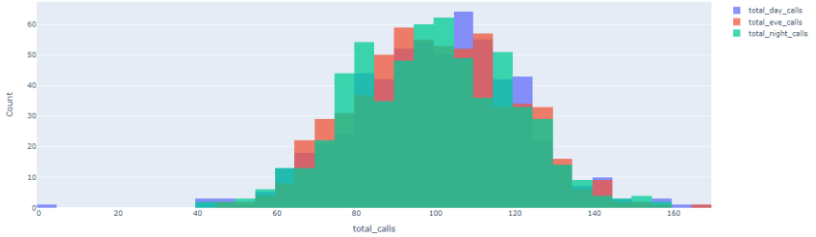
numeric

BINAR

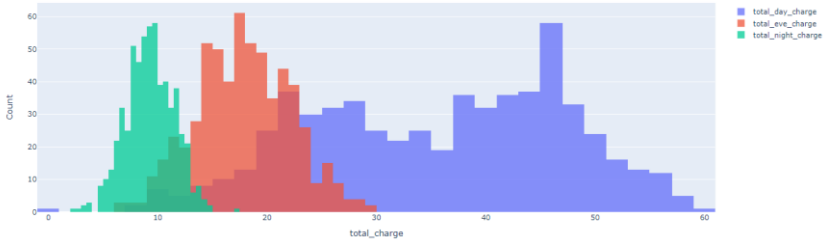
churn

total minutes (day, eve, night)

total calls (day, eve, night)



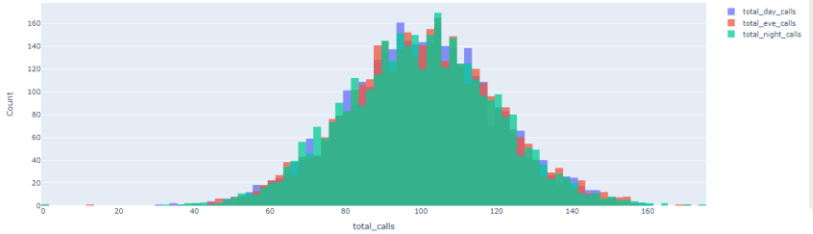
total charge (day, eve, night)



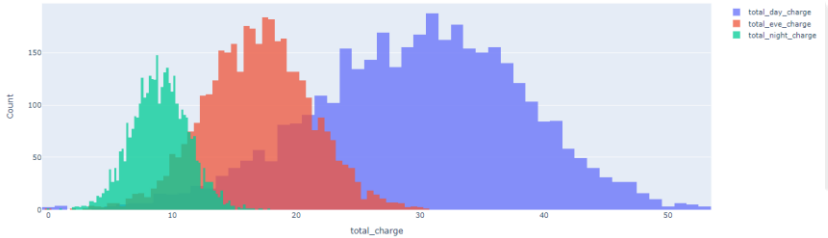
no-churn

total minutes (day, eve, night)

total calls (day, eve, night)

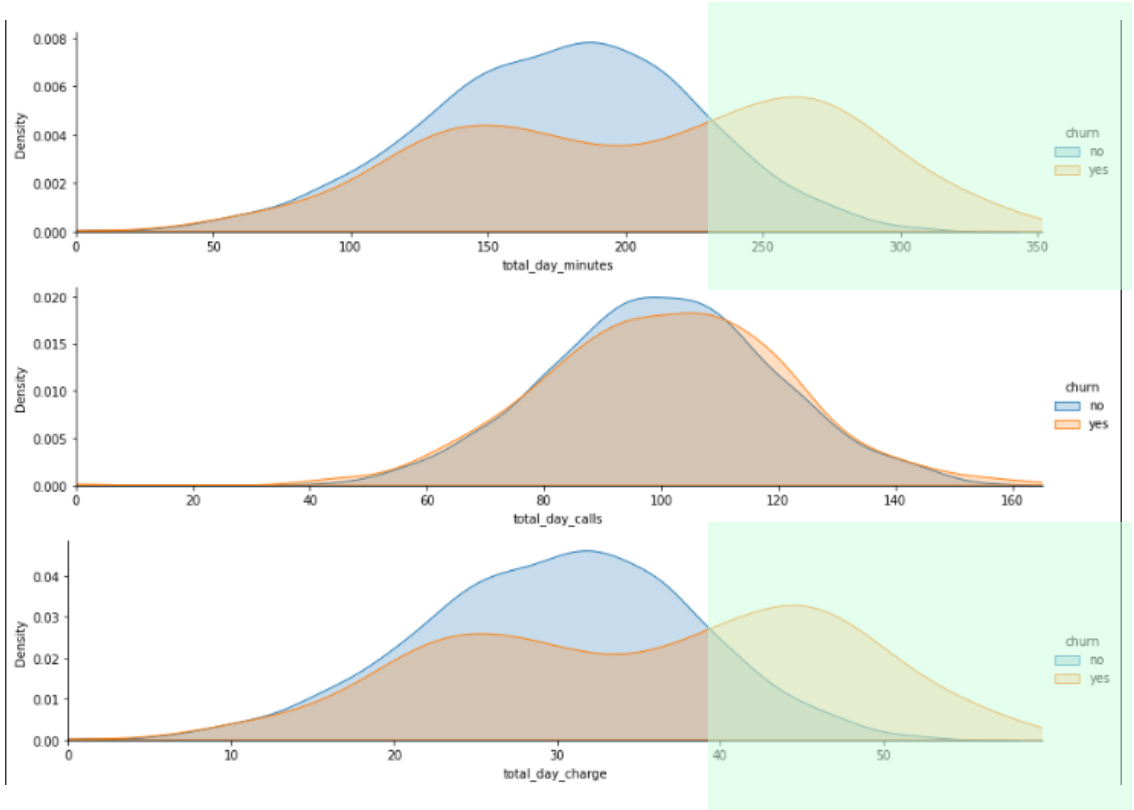


total charge (day, eve, night)



Eksplorasi Data

waktu: **day calls**



Grafik memperlihatkan pelanggan yang menggunakan layanan di waktu-**day** dengan **total menit > 225** memiliki kecendrungan melakuan-**churn** (**beralih**)

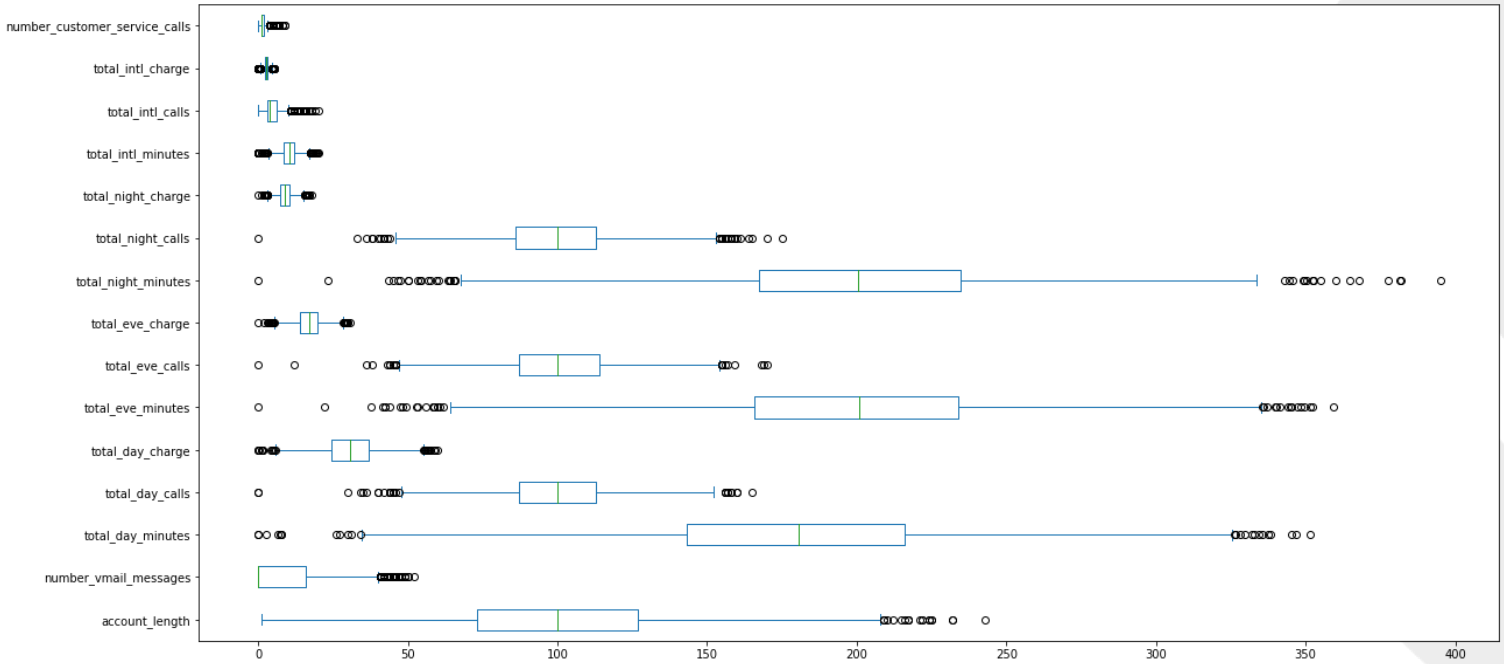
BINAR

Data Pre-processing

Data preprocessing (outlier)

tidak ada data kosong
tidak ada duplikasi

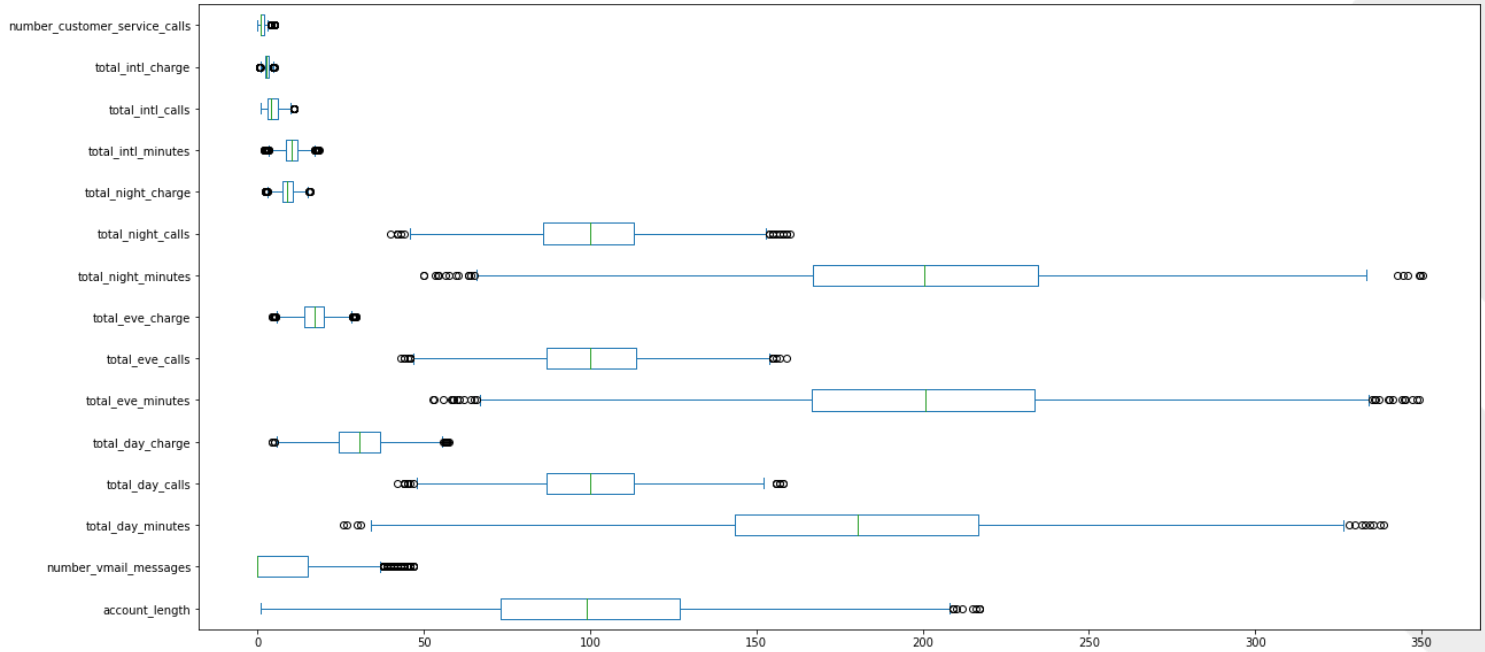
Menggunakan zscore (filter zscore<3)



Data preprocessing (outlier)

Menggunakan zscore (filter zscore<3)

4250 → 4031



Data preprocessing (feature engineering)

BINAR

```
# categorical feature
# yes & no to 1 & 0
df_train_clean['voice_mail_plan'] = df_train_clean['voice_mail_plan']\
                                        .map({'yes': 1, 'no': 0})
df_train_clean['international_plan'] = df_train_clean['international_plan']\
                                         .map({'yes': 1, 'no': 0})
df_train_clean['churn'] = df_train_clean['churn']\
                            .map({'yes': 1, 'no': 0})

# onehot area_code
df_train_clean = pd.get_dummies(data=df_train_clean, columns=['area_code'])
```

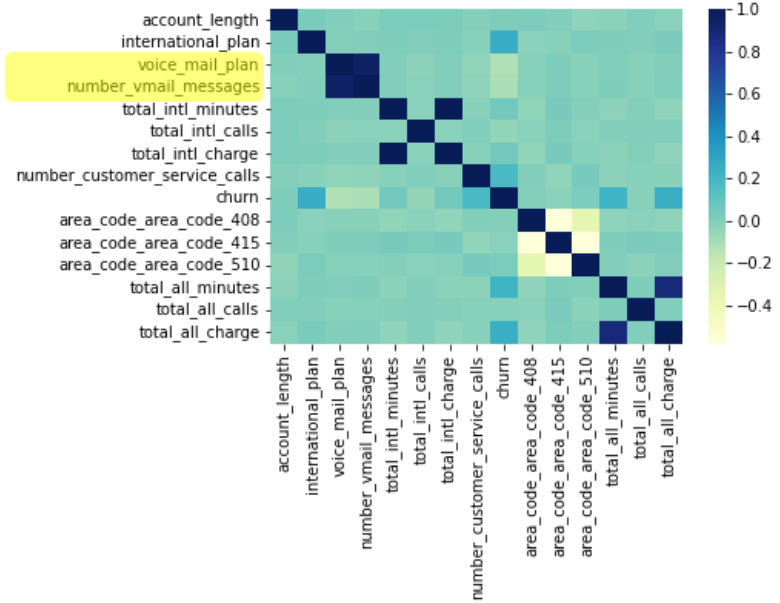
Data preprocessing (feature engineering)

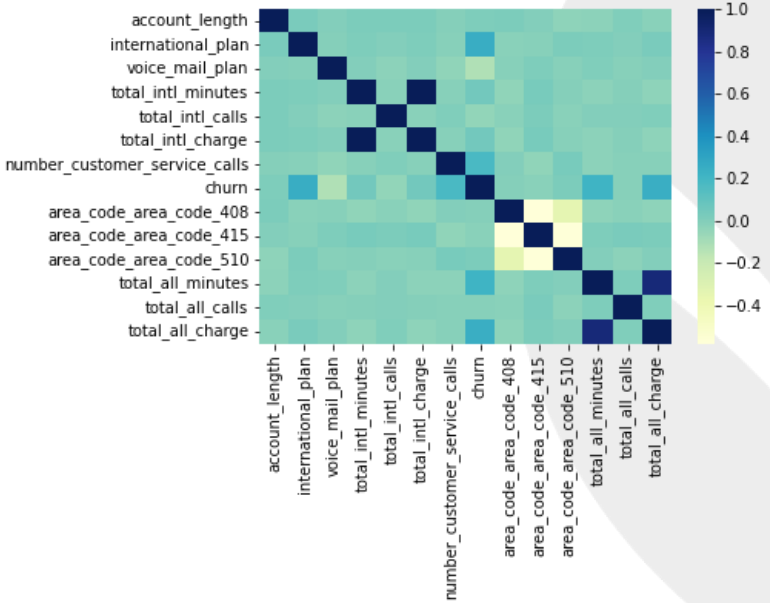
BINAR

```
# numerical feature
df_train_clean['total_all_minutes'] = df_train_clean['total_day_minutes'] +\
                                      df_train_clean['total_eve_minutes'] +\
                                      df_train_clean['total_night_minutes']
df_train_clean['total_all_calls'] = df_train_clean['total_day_calls'] +\
                                    df_train_clean['total_eve_calls'] +\
                                    df_train_clean['total_night_calls']
df_train_clean['total_all_charge'] = df_train_clean['total_day_charge'] +\
                                     df_train_clean['total_eve_charge'] +\
                                     df_train_clean['total_night_charge']
```

Data preprocessing (feature selection)

drop number_vmail_messages





Data preprocessing (feature selection)

BINAR

```
international_plan      2
voice_mail_plan         2
churn                   2
area_code               3
state                  51
dtype: int64
```

unique

drop

```
#drop columns
df_train_clean.drop(columns=[ 'state','total_day_charge', 'total_eve_charge',
                              'total_night_charge', 'total_day_calls',
                              'total_eve_calls', 'total_night_calls',
                              'total_day_minutes', 'total_eve_minutes',
                              'total_night_minutes'], inplace=True)
```

Fitur yang digunakan 14

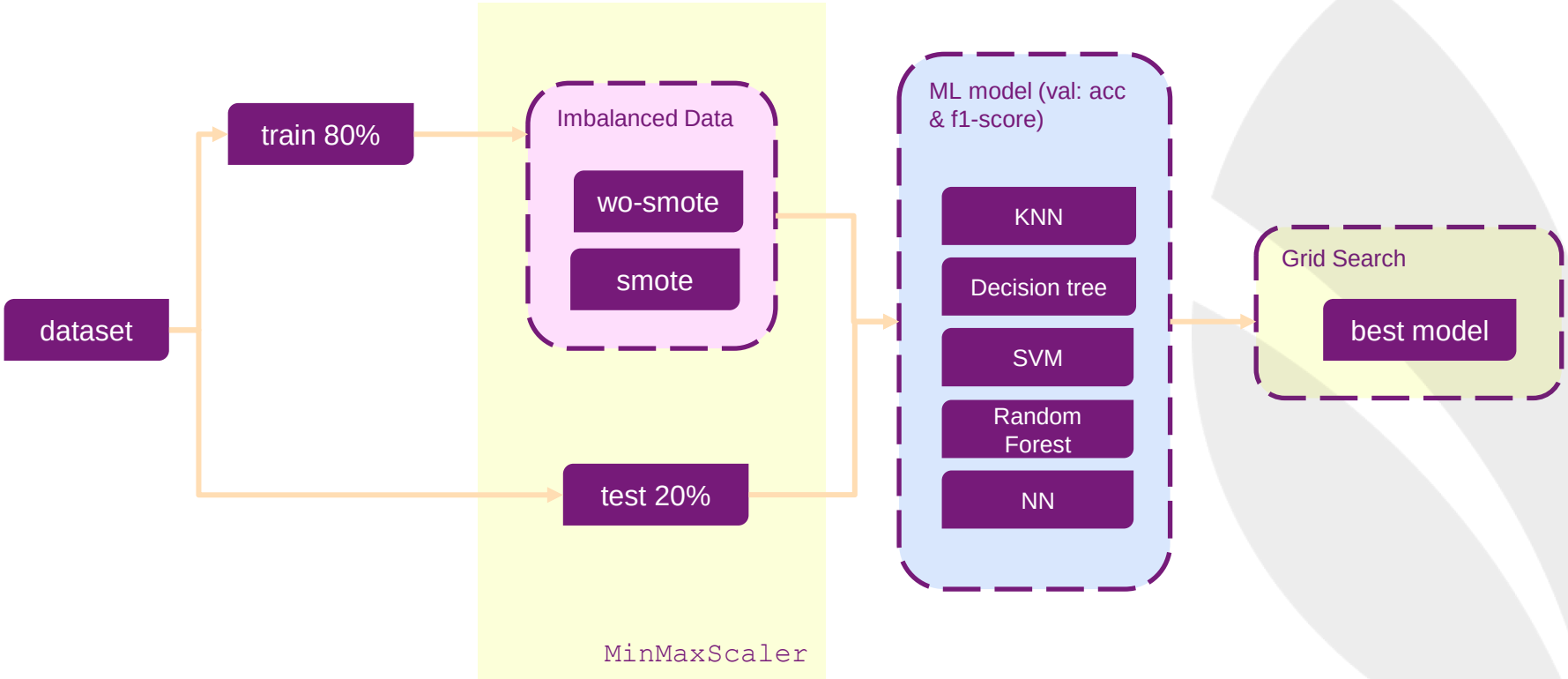
```
df_train_clean.keys()
```

```
Index(['account_length', 'international_plan', 'voice_mail_plan',
       'total_intl_minutes', 'total_intl_calls', 'total_intl_charge',
       'number_customer_service_calls', 'churn', 'area_code_area_code_408',
       'area_code_area_code_415', 'area_code_area_code_510',
       'total_all_minutes', 'total_all_calls', 'total_all_charge'],
      dtype='object')
```

Modeling

Modeling

BINAR



BINAR

TRAIN

wo-smote

```
y_train.value_counts()
```

```
0    2786
1     439
Name: churn, dtype: int64
```

smote

```
from imblearn.over_sampling import SMOTE
smote = SMOTE(k_neighbors=5)
X_train_b, y_train_b = smote.fit_resample(X_train, y_train)
print(y_train_b.value_counts())
```

```
0    2786
1    2786
Name: churn, dtype: int64
```

oversampling

TEST

```
y_test.value_counts()
```

```
0    701
1    105
Name: churn, dtype: int64
```

Modeling

KNN
DecisionTree
RandomForest
SVM

```
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, f1_score
models = {
    'KNeighborsClassifier': KNeighborsClassifier(),
    'DecisionTreeClassifier': DecisionTreeClassifier(),
    'RandomForestClassifier': RandomForestClassifier(),
    'SVC': SVC(kernel='linear'),
}
dict_non_f1= {}
dict_non_acc = {}
dict_non_model = {}
fig, axs = plt.subplots(len(models), 1, figsize=(10, len(models)*5))
for model_name, i in zip(models, range(len(models))):
    # Load the model obj, fit, predict and calc f1-score
    model = models[model_name]
    # normalisasi data
    scaler = MinMaxScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)
    model.fit(
        X_train_scaled,
        y_train
    )
    y_pred=model.predict(X_test_scaled)
    f1=f1_score(y_test, y_pred, average='weighted')
    acc=accuracy_score(y_test, y_pred)

    # Load the result into the created dict
    dict_non_f1[model_name]=f1
    dict_non_acc[model_name]=acc
    dict_non_model[model_name]=model

    cm=confusion_matrix(y_test, y_pred)

    # Plot results
    axs[i].set_title(f'{model_name}: acc={np.round(acc, 5)} -- f1={np.round(f1, 5)}')
    sns.heatmap(ax=axs[i], data=cm, annot=True, fmt='g', cbar=False)
plt.show()
```

Modeling

NN

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow_addons.metrics import F1Score

model_nn = Sequential()
model_nn.add(Dense(128, input_dim=X_train_scaled.shape[1],
                   activation='relu')),
model_nn.add(Dense(256, activation='relu')),
model_nn.add(Dropout(0.2)),
model_nn.add(Dense(1, activation='sigmoid')),

model_nn.compile(loss='binary_crossentropy',
                 optimizer=Adam(learning_rate=1e-2),
                 metrics=['accuracy',
                          F1Score(num_classes=1,
                                  average='weighted',
                                  threshold=0.5)])
model_nn.summary()
```

```
from keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(min_delta = 0.001, mode = 'max',
                               monitor='val_f1_score', patience = 15,
                               restore_best_weights=True)
callbacks = [early_stopping]
```

```
history = model_nn.fit(X_train_scaled, y_train,
                       validation_data=(X_test_scaled, y_test),
                       epochs=500, verbose=2, callbacks=[callbacks])
```

Modeling

hasil **data-test(20%)**

```
y_test.value_counts()

0    701
1    105
Name: churn, dtype: int64
```

BINAR

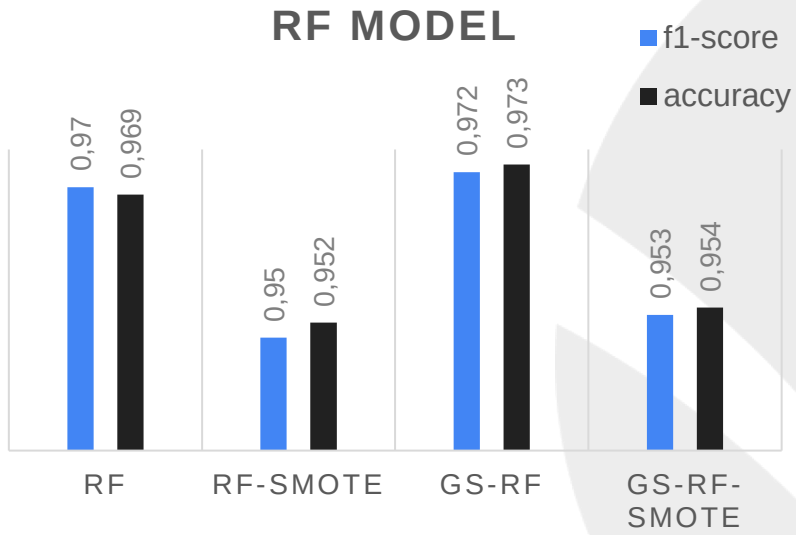
Nama model	F1-score	ACC	TN	FN	TP	FP
knn	0.889	0.905	693	8	37	68
svm	0.809	0.869	701	0	0	105
decisionTree	0.954	0.954	684	17	85	20
randomForest	**0.970**	**0.969**	**698**	**3**	**84**	**21**
neuralNetwork	0.964	0.965	697	4	81	24
knn-smote	0.865	0.859	635	66	58	47
svm-smote	0.855	0.859	635	66	54	51
decisionTree-smote	0.898	0.888	628	73	88	17
randomForest-smote	0.950	0.952	687	14	80	25
neuralNetwork-smote	0.948	0.949	686	15	79	26

Nilai FP lebih tinggi dari TP sehingga
Prediksi model condong ke salah satu class (mengarah ke churn-**no**)

smote berperan baik dalam mengatasi imbalance data pada model knn & svm

Modeling (grid search)

```python
from sklearn.model_selection import GridSearchCV, KFold
from sklearn.ensemble import RandomForestClassifier

#RF GridSearch
param_grid = {
    'bootstrap': [True],
    'n_estimators': [128, 512, 1024],
    'max_depth' : [8, 16, 32, 64, 128]
}

rf = RandomForestClassifier()
rf_gs = GridSearchCV(
    estimator=rf,
    scoring='f1_weighted',
    param_grid=param_grid,
    cv=KFold(n_splits=10),
    verbose=2,
    n_jobs=-1)

rf_gs.fit(X_train_scaled, y_train)
print(rf_gs.best_params_)
```



wo-smote

```
Fitting 10 folds for each of 15 candidates, totalling 150 fits
{'bootstrap': True, 'max_depth': 16, 'n_estimators': 1024}
```

with-smote

```
Fitting 10 folds for each of 15 candidates, totalling 150 fits
{'bootstrap': True, 'max_depth': 64, 'n_estimators': 512}
```

Penerapan grid search pada model dapat menambah sedikit poin

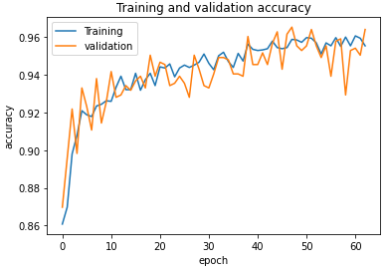
Poin GS-RF sedikit diatas RF dimana
f1-score 0,004 lebih tinggi dan
acc 0,002 lebih tinggi

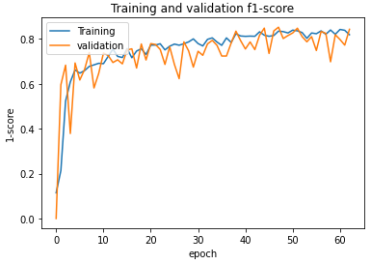
Modeling

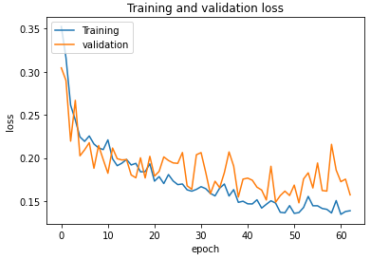
NN

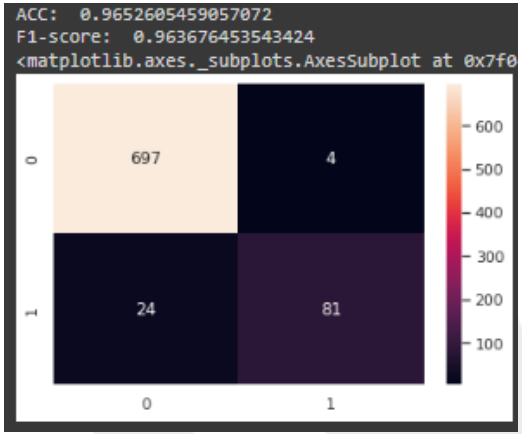
hasil **data-test(20%)**

wo-smote

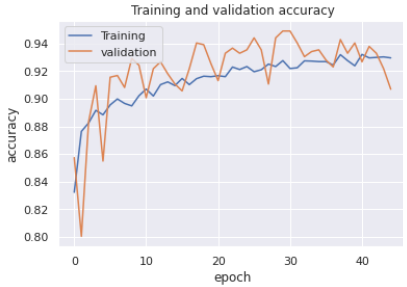


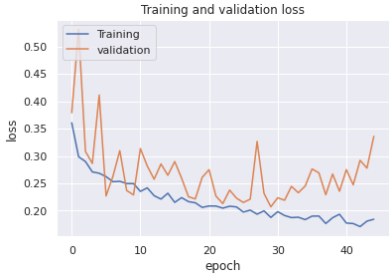


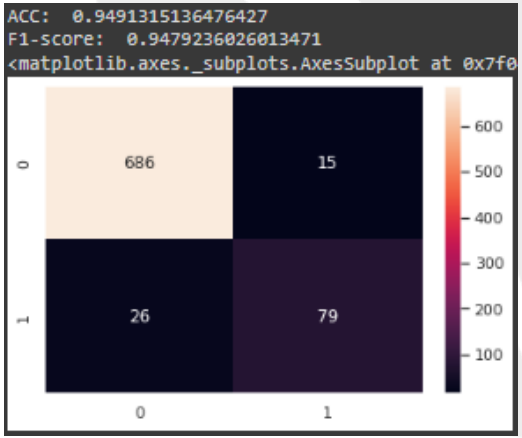




smote







Penerapan SMOTE pada KNN dan SVM berperan baik, berbeda dengan tree-base (Desicion Tree dan RF)

Model RF tanpa SMOTE menjadi model terbaik (acc: 0.97 & f1-scor: 0.969)

Penerapan hyperparameter tuning pada model RF menghasilkan sedikit peninggkatan (acc: 0.973 & f1-score: 0.972)

Saran:

Menggunakan metode lain untuk mengatasi data tidak seimbang

Mencari nilai k-neighbors optimum pada oversampling (SMOTE)

Melakukan hyperparameter tuning pada beberapa model

Terimakasih!