Rakamin Academy

DATA GEEKS

Data Geeks

Dokumen Laporan Final Project

(dipresentasikan setiap sesi mentoring)

Import Packages

```
# Melakukan import library
import warnings
warnings.filterwarnings('ignore')

import numpy as np
import pandas as pd
import seaborn as sns
from scipy import stats
import matplotlib.pyplot as plt
from matplotlib import rcParams
%matplotlib inline
from sklearn.preprocessing import MinMaxScaler, StandardScaler, RobustScaler
from scipy.stats import boxcox
from imblearn import under_sampling, over_sampling
import gdown
from sklearn.model_selection import train_test_split

from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, roc_curve, confusion_mat
from sklearn.experimental import enable_halving_search_cv
from sklearn.model_selection import cross_validate, RandomizedSearchCV, GridSearchCV, HalvingGridSearchCV
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression, Ridge, Lasso, ElasticNet
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier, reset_parameter, LGBMClassifier

import shap

from scipy.stats import randint as sp_randint
from scipy.stats import uniform as sp_uniform

print('numpy version : ',np.__version__)
print('pandas version : ',pd.__version__)
print('seaborn version : ',sns.__version__)
```

```
numpy version :  1.23.5
pandas version :  2.0.2
seaborn version :  0.12.2
```

→ Import library yang dibutuhkan

```
sns.set(rc={'figure.figsize':(20.7,8.27)})
sns.set_style("whitegrid")
sns.color_palette("dark")
plt.style.use("fivethirtyeight")
```

→ Mengatur setting visualisasi secara global.

Load Data

```
# Melakukan import csv
df = pd.read_csv("train.csv")
df.sample(5)
```

	id	Gender	Age	Driving_License	Region_Code	Previously_Insured	Vehicle_Age	Vehicle_Damage	Annual_Premium	Poli
230397	230398	Male	48	1	8.0	0	1-2 Year	Yes	36694.0	
340418	340419	Female	26	1	8.0	1	< 1 Year	No	42370.0	
151050	151051	Male	33	1	29.0	0	< 1 Year	Yes	2630.0	
280196	280197	Male	72	1	28.0	1	1-2 Year	No	64997.0	
291720	291721	Male	41	1	28.0	1	1-2 Year	No	43987.0	

↓

Load dataset (.csv)
Melihat 5 buah samples

Pengelompokan dan Penentuan Target

```
# Pengelompokan kolom berdasarkan jenisnya
nums = ['int64', 'int32', 'int16', 'float64', 'float32', 'float16']
nums = df.select_dtypes(include=nums)
nums.drop(columns=['id'],inplace=True)
nums = nums.columns
cats = ['Gender','Vehicle_Age','Vehicle_Damage']
```

Target dari modelling classification ini adalah kolom **Response**.

Mengelompokkan kolom berdasarkan tipe datanya
baik numerik/kategorikal.

1. Data Preprocessing

A. Handling Missing Value

```
# Melihat jumlah baris dan kolom
test = df.shape[0]

# Mengecek missing value ditiap fitur
missing_values_count = df.isnull().sum()
missing_values_count
```

```
id                    0
Gender                0
Age                   0
Driving_License       0
Region_Code           0
Previously_Insured    0
Vehicle_Age           0
Vehicle_Damage        0
Annual_Premium        0
Policy_Sales_Channel  0
Vintage               0
Response              0
dtype: int64
```

Dari 12 kolom tidak ditemukan nilai kosong.

Menggunakan function **.isnull() / .isna()** untuk melihat apakah ada kolom yang memiliki nilai kosong.

1. Data Preprocessing

B. Handling Duplicates Value

```
df.duplicated().sum()
```

```
0
```

```
df.duplicated(subset=['id'],keep=False).sum()
```

```
0
```

Tidak terdapat data duplikasi pada dataset

1. Data Preprocessing

C. Handling Outliers

```
# membuat function untuk mencari IQR
d = df.shape[0]
a = ['Annual_Premium']
def find_limit(df, variables) :
    q1 = df[variables].quantile(0.25)
    q3 = df[variables].quantile(0.75)
    iqr = q3 - q1
    lower_limit = q1 - (1.5 * iqr)
    upper_limit = q3 + (1.5 * iqr)
    return lower_limit, upper_limit
```

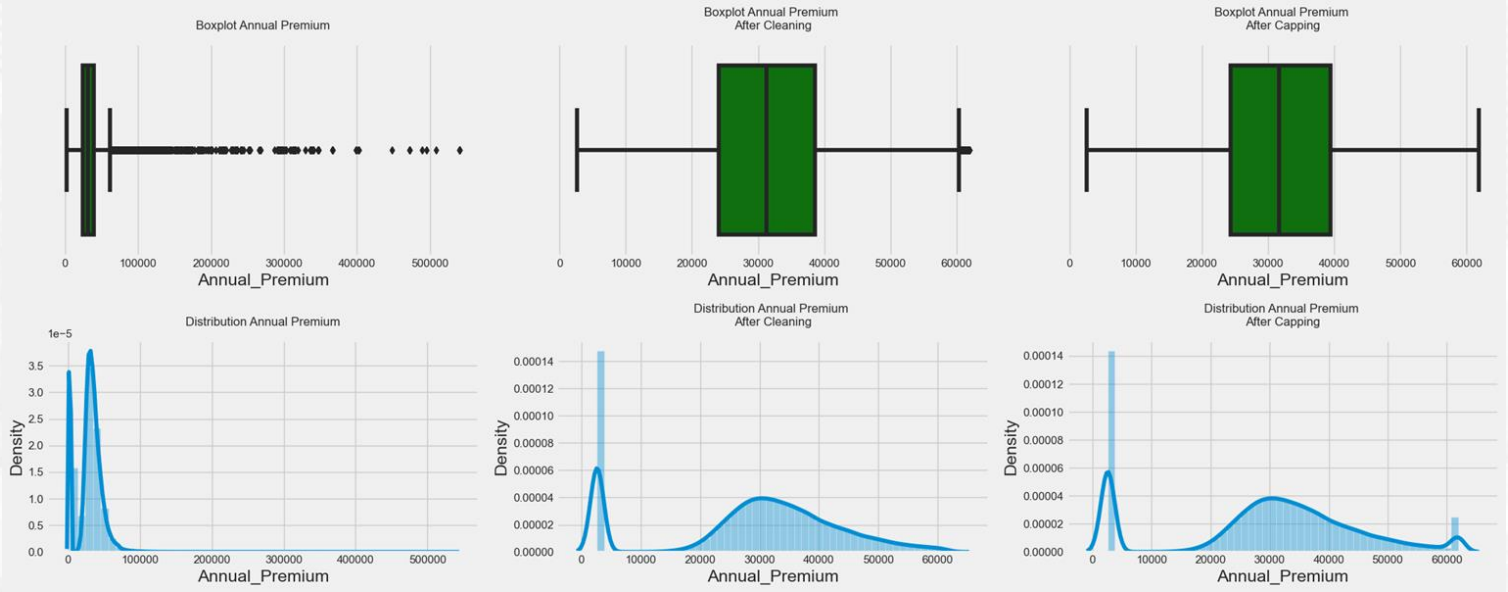
```
# Membuang value outliers
df_clean = df.copy()

for i in a :
    lower, upper = find_limit(df_clean, i)
    df_clean = df_clean[~((df_clean[i] < lower) | (df_clean[i] > upper))]

df_clean = df_clean.reset_index(drop = True)

print(f'Jumlah baris sebelum memfilter outlier: {d}')
print(f'Jumlah baris setelah memfilter outlier: {len(df_clean)}')
print(f'Sebanyak {d-len(df_clean)} rows dihapuskan atau {round(((d-len(df_clean))/d)*100,2)} %')
```

```
Jumlah baris sebelum memfilter outlier: 381109
Jumlah baris setelah memfilter outlier: 370789
Sebanyak 10320 rows dihapuskan atau 2.71 %
```

1. Data Preprocessing

C. Handling Outliers

```
# Capping : Mengubah outlier value menjadi upper (atau) lower
df_cap = df.copy()

for i in a :
    lower, upper = find_limit(df_cap, i)
    df_cap.loc[(df_cap[i] > upper), i] = upper
    df_cap.loc[(df_cap[i] < lower), i] = lower
```



Diputuskan untuk **tetap menggunakan dataframe df** karena, kolom Annual_Premium merupakan hal yang **normal jika terdapat outliers** sehingga t**idak dilakukan penghapusan outliers**. Hal ini juga didasarkan dengan pertimbangan **pembuatan model yang robust terhadap outliers**.

1. Data Preprocessing

D. Feature Encoding

```
for i in ['Gender']:
    onehots = pd.get_dummies(df['Gender'], prefix='Gen')
    df = df.join(onehots)

df[['Gen_Female','Gen_Male']] = df[['Gen_Female','Gen_Male']].astype(int)
```

```
mapping_damage = {'Yes' : 1, 'No' : 0}
df['Vehicle_Damage'] = df['Vehicle_Damage'].map(mapping_damage)
```

```
mapping_age = {'> 2 Years' : 2, '1-2 Year' : 1, '< 1 Year' : 0}
df['Vehicle_Age'] = df['Vehicle_Age'].map(mapping_age)
```

```
df.head(2)
```

	id	Gender	Age	Driving_License	Region_Code	Previously_Insured	Vehicle_Age	Vehicle_Damage	Annual_Premium	Policy_Sales_Cl
0	1	Male	44	1	28.0	0	2	1	40454.0	
1	2	Male	76	1	3.0	0	1	0	33536.0	

Mengubah **Vehicle_Damage** ke integer dalam = 0: Kendaraan customer belum pernah rusak, 1: Kendaraan customer sudah pernah rusak, serta **Vehicle_Age** dam 0: < 1 Year, 1: 1-2 Years, 2: > 2 Years. Serta **Gender** dengan *One Hot Encoding*. Melakukan konversi ke angka mulai dari 0 untuk memudahkan kerja machine learning. Mengubah kolom dengan data type bool ke int agar lebih mudah diproses oleh model.

1. Data Preprocessing

E. Class Imbalance

```
# Melihat berapa dejarat ketimpangan pada class

for i in range(len(df['Response'].value_counts())):
    a = round(df['Response'].value_counts()[i]/df.shape[0]*100,2)
    print(f'{a}%')

print('Degree of Imbalance dari data ini termasuk pada Moderate Imbalance')
```

```
87.74%
12.26%
Degree of Imbalance dari data ini termasuk pada Moderate Imbalance
```

↓

Melihat berapa derajat ketimpangan pada class

Penanganan Class Imbalance dilakukan dengan ***oversampling*** dan ***undersampling*** dengan pertimbangan agar data tidak cenderung bias, dimana selisih antara kedua value 0 dan 1 lebih dari 50% sehingga jika dilakukan oversampling tidak menjamin akan adanya peningkatan performansi machine learning, namun dibutuhkan pula oversampling agar data tidak underfit.

1. Data Preprocessing

E. Class Imbalance

```
# Membuat kolom baru untuk melihat Class 'Yes'

df['Res_class'] = df['Response']==1
df['Res_class'].value_counts()
```

```
Res_class
False    334399
True      46710
Name: count, dtype: int64
```

↓

Membuat kolom baru untuk melihat memisahkan class Yes sebagai True dan No sebagai False

```
# Memisahkan dataframe tanpa Response dan Res_class dan hanya Res_class
X = df[[col for col in df.columns if (str(df[col].dtype) != 'object') and col not in ['Response', 'Res_class']]]
y = df['Res_class'].values
print(X.shape)
print(y.shape)
```

```
(381109, 12)
(381109,)
```

↓

Memisahkan dataframe utuh tanpa Response dan Res_class (X) dan dataframe yang hanya berisikan value Res_class (y)

1. Data Preprocessing

E. Class Imbalance

Melakukan oversampling SMOTE dengan sampling strategy 30% dan memasukkannya ke dataframe df.

```
X_over_SMOTE, y_over_SMOTE = over_sampling.SMOTE(sampling_strategy=0.3, random_state=42).fit_resample(X, y)
```

```
pd.Series(y_over_SMOTE).value_counts()
```

```
False    334399
True     100319
Name: count, dtype: int64
```

```
X_over_SMOTE['Response'] = y_over_SMOTE.astype(int)
df = X_over_SMOTE.copy()
```

Mengulangi proses pemisahan sebelum undersampling

```
# Memisahkan dataframe tanpa Response dan Res_class dan hanya Res_class
df['Res_class'] = df['Response']==1
df['Res_class'].value_counts()
X2 = df[[col for col in df.columns if (str(df[col].dtype) != 'object') and col not in ['Response', 'Res_class']]]
y2 = df['Res_class'].values
print(X2.shape)
print(y2.shape)
```

```
(434718, 12)
(434718,)
```

1. Data Preprocessing

E. Class Imbalance

Melakukan undersampling dan memasukkannya ke dataframe df.

```
X_under, y_under = under_sampling.RandomUnderSampler(sampling_strategy=1, random_state=42).fit_resample(X2, y2)
```

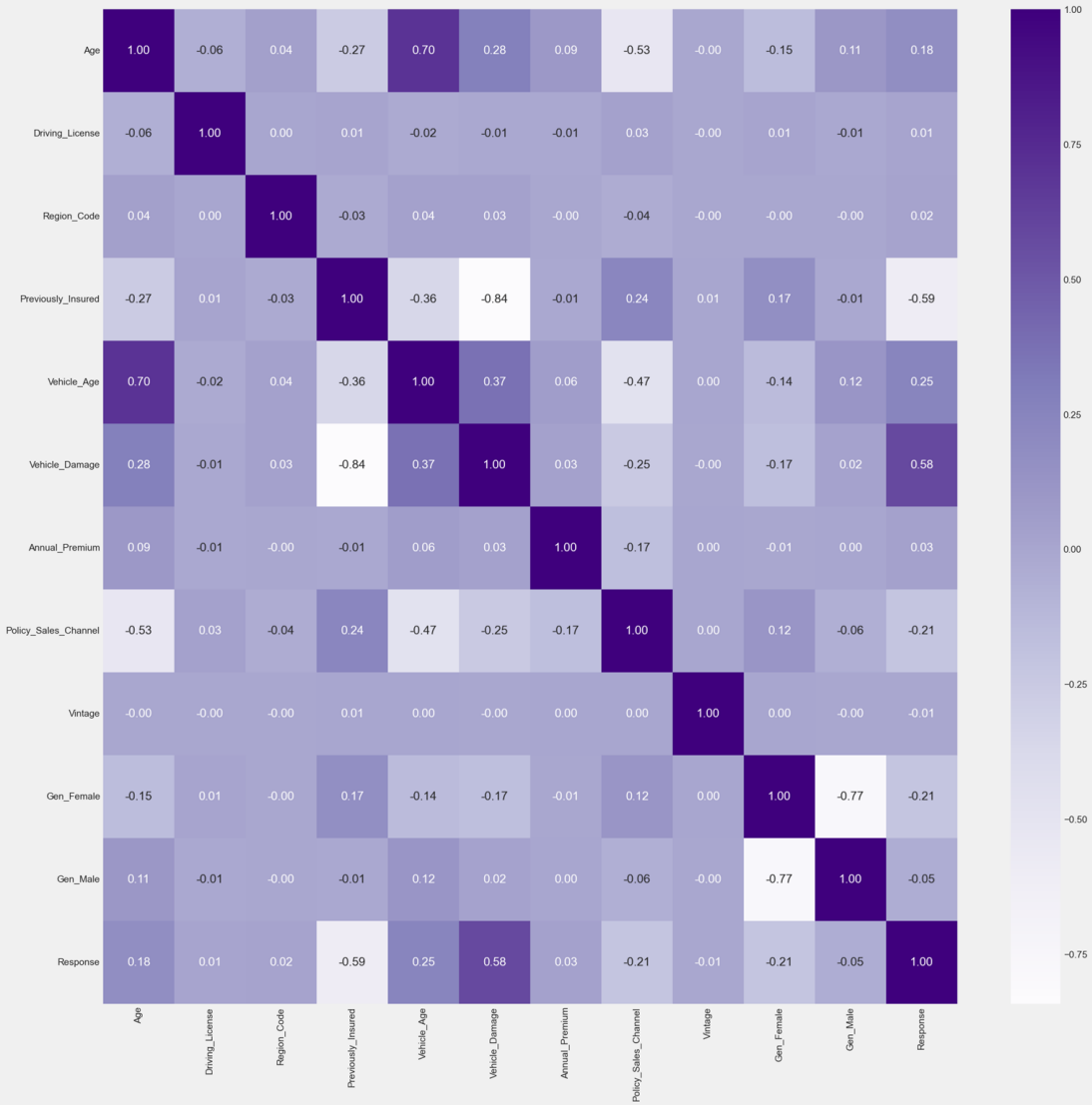
```
print('Original')
print(pd.Series(y).value_counts())
print('\n')
print('OVERSAMPLING SMOTE & UNDERSAMPLING')
print('')
print(pd.Series(y_under).value_counts())
```

```
Original
False    334399
True      46710
Name: count, dtype: int64


OVERSAMPLING SMOTE & UNDERSAMPLING

False    100319
True     100319
Name: count, dtype: int64
```

```
X_under['Response'] = y_under.astype(int)
df = X_under.copy()
```

2. Feature Engineering

A. Feature Selection



Dari heatmap dapat didapatkan insight bahwa, **Age** dan **Vehicle_Age** merupakan kolom redundant sehingga diputuskan untuk tidak menggunakan pada kolom **Age** dengan pertimbangan kolom **Age** memiliki korelasi lebih kecil dibandingkan **Vehicle_Age**.

2. Feature Engineering

B. Feature Extraction

```
# Membuat fitur kategori baru dari Age dengan mengelompokkan YoungAdults 17 - 30, MiddleAged 31-45, OldAdults > 45
Age_Group = []
for i in df['Age']:
    if 17 <= i <=30:
        Age_Group.append(0)
    elif 31 <= i <= 45:
        Age_Group.append(1)
    else:
        Age_Group.append(2)

df['Age_Group'] = Age_Group
df
```

Membuat fitur kategori baru dari kolom Age dengan mengelompokan YoungAdults rentang usia 17 - 30 tahun, kemudian MiddleAged rentang usia 31 - 45 tahun dan OldAdults dengan rentang usia diatas 45 tahun

2. Feature Engineering

B. Feature Extraction

```
# Melakukan ekstrasi dari kolom Annual_Premium dengan LowPremium < 24406, MediumPremium 24406 - 61892.4 dan HighPremium >
c = []

for i in df['Annual_Premium']:
    if i <=  24405:
        c.append(0)
    elif 24406 <= i <= 61892.4:
        c.append(1)
    else:
        c.append(2)

df['Premium_cat'] = c
df
```

Melakukan ekstraksi dari kolom Annual_Premium dengan dengan kategori LowPremium < 24406, MediumPremium rentang 24406 - 61892.4, dan HighPremium > 61892.4

2. Feature Engineering

B. Feature Extraction

Melakukan ekstraksi dari kolom Policy_Sales_Channel berdasarkan value_count

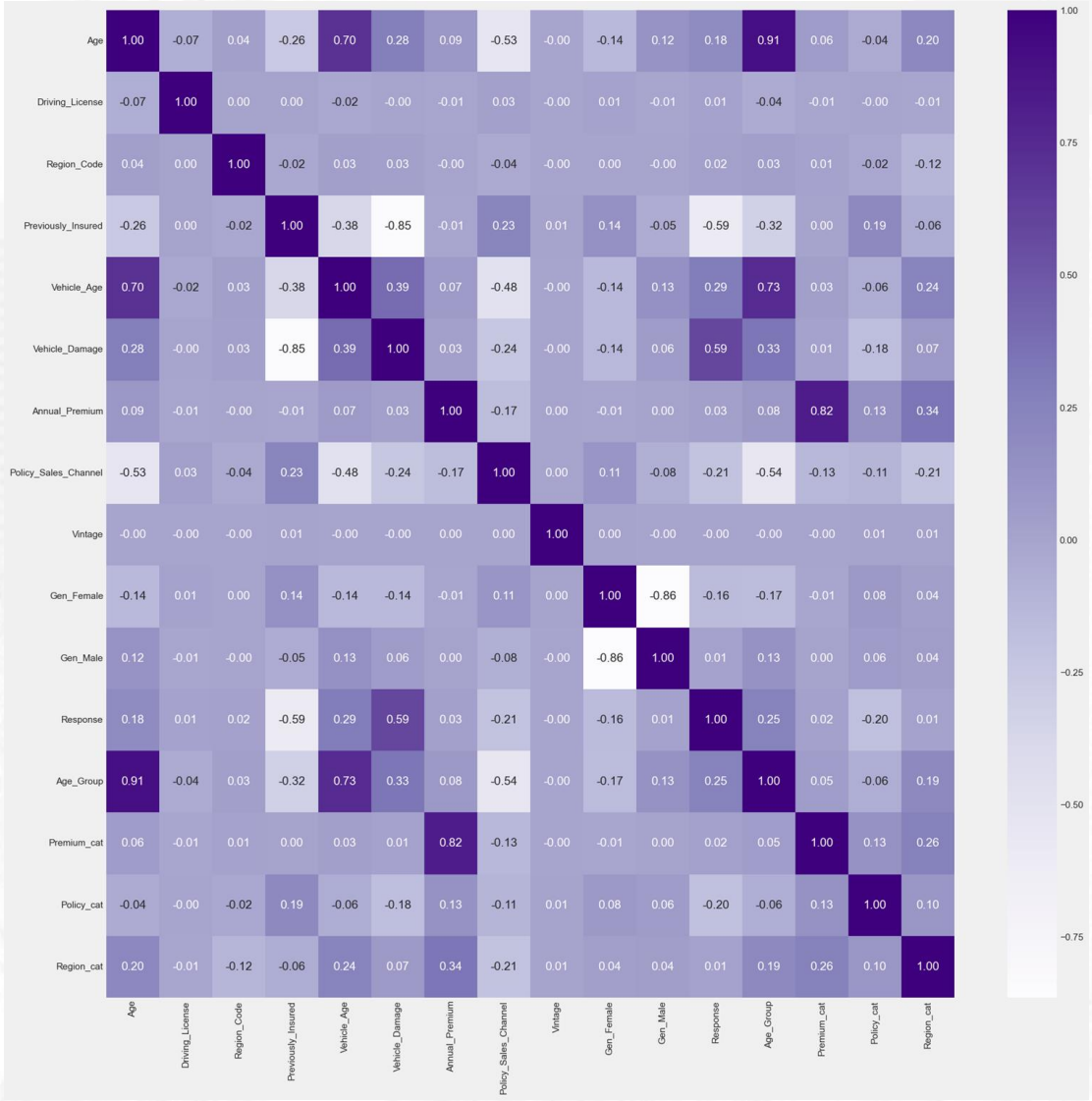
```
# Melakukan ekstrasi kolom Policy_Sales_Channel berdasarkan value_counts
p = df['Policy_Sales_Channel'].value_counts()
Policy_cat = p.apply(lambda x : 3 if x>=16455 else 2 if 8227<x<16455 else 1 if x<=8227 else 0)
Policy_cat = dict(zip(Policy_cat.keys(),Policy_cat.values))
df['Policy_cat'] = df['Policy_Sales_Channel'].map(Policy_cat)
df
```

Melakukan ekstraksi dari kolom Region_Code berdasarkan value_counts

```
# Melakukan ekstrasi kolom Region_Code berdasarkan value_counts
r = df['Region_Code'].value_counts()
Region_cat = r.apply(lambda x : 2 if x > 8339 else 1 if 2611<=x<=8339 else 0)
Region_cat = dict(zip(Region_cat.keys(),Region_cat.values))
df['Region_cat'] = df['Region_Code'].map(Region_cat)
df
```

2. Feature Engineering

B. Feature Extraction



Dari hasil *feature extraction* didapatkan insight bahwa **Policy_cat** tidak memiliki korelasi dengan **Response**, sedangkan **Age_Group** dan **Premium_cat** memiliki korelasi *positive*.

2. Feature Engineering

C. New Features

1. **Premium_Per_Channel**, untuk menghitung dan memberi insight baru mengenai total premium dari berbagai **Policy_Sales_Channel** dengan begitu pengelompokkan Channel dapat dilakukan berdasarkan **Annual_Premium**.
2. **Vintage_Group**, feature baru yang mengubah feature **Vintage** menjadi kategori dengan range tertentu dimana diartikan menjadi New (baru bergabung), Intermediate (sudah bergabung cukup lama), Long-term (sudah bergabung lama).
3. **Not_Insured_and_Damaged**, kolom yang berisikan nilai 1 jika kolom **Previously_Insured** memiliki value 0 dan **Vehicle_Damage** memiliki value 1.
4. **Channel_Response_Rate**, merupakan rate respon dari tiap channel dimana mengindikasikan seberapa efektif suatu channel untuk mendapatkan jawaban 'Yes' dari sini juga dapat dilakukan pengelompokkan Channels yang memiliki rate tinggi.

Data Preprocessing

Feature Transformation

Data memiliki karakteristik mayoritas sudah berbentuk Gaussian namun ada beberapa yang masih belum tersebar normal.

Melakukan split test and train serta penggabungan kembali *features* dan target.

```
# Mengkelompokkan kolom-kolom yang akan menjadi features serta target
x = ['int64', 'int32', 'int16', 'float64', 'float32', 'float16']
x = df.select_dtypes(include=x)
x.drop(columns=['Response'],inplace=True)
y = df['Response']
```

```
# Melakukan split test and train
Xtrain, Xtest, ytrain, ytest = train_test_split(x, y, test_size=1/3, random_state=42)

# Menggabungkan kembali features dan target untuk membuat dataframe test dan train

df_train = Xtrain.join(ytrain)
df_test  = Xtest.join(ytest)
```

Data Preprocessing

Feature Transformation

Dari seluruh kolom yang bukan merupakan kolom kategori adalah **Age**, **Annual_Premium** dan **Vintage** sehingga hanya ketiga kolom ini yang dilakukan transformasi.

Melakukan log *transformation* untuk kolom **Age** yang *positive skewed* dan mencoba StandardScaler dan Boxcox pada kolom **Annual_Premium** dan **Vintage**.

```
f_log = ['Age']
for i in f_log:
    df_train['log_'+i] = np.log(df_train[i] + (df_train[df_train[i] > 0][i].min() / 2))
    df_test['log_'+i] = np.log(df_test[i] + (df_test[df_test[i] > 0][i].min() / 2))
```

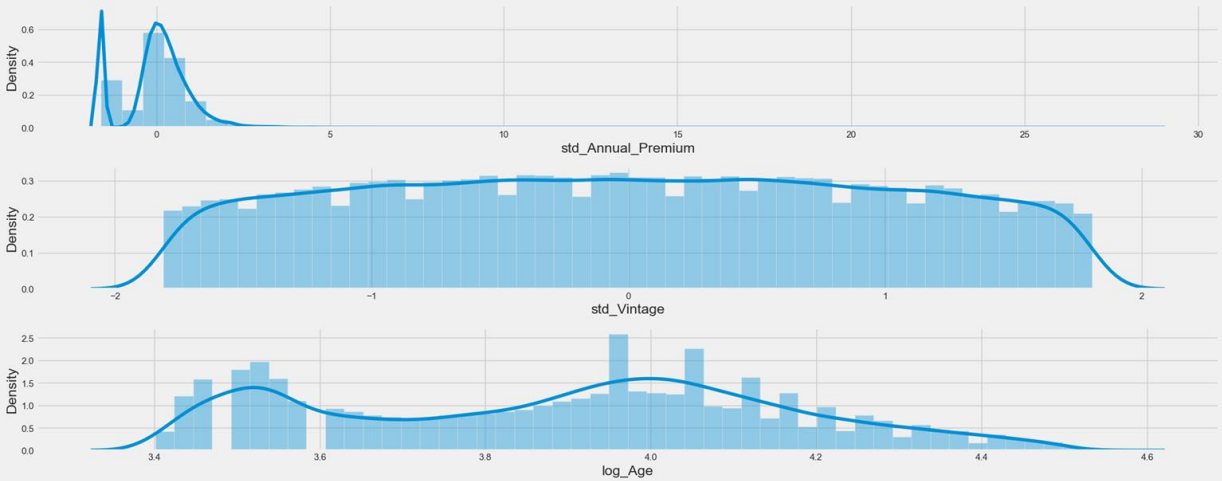
```
sscale = ['Annual_Premium','Vintage']
scaler = StandardScaler()
for i in sscale:
    scaler.fit(df_train[[i]])
    scaler.fit(df_test[[i]])
    df_train['std_'+i] = scaler.transform(df_train[[i]])
    df_test['std_'+i] = scaler.transform(df_test[[i]])
```

```
sscale = ['Annual_Premium','Vintage']
for i in sscale:
    df_train['box_'+i],_ = boxcox(df_train[i]+1)
    df_test['box_'+i],_ = boxcox(df_test[i]+1)
```

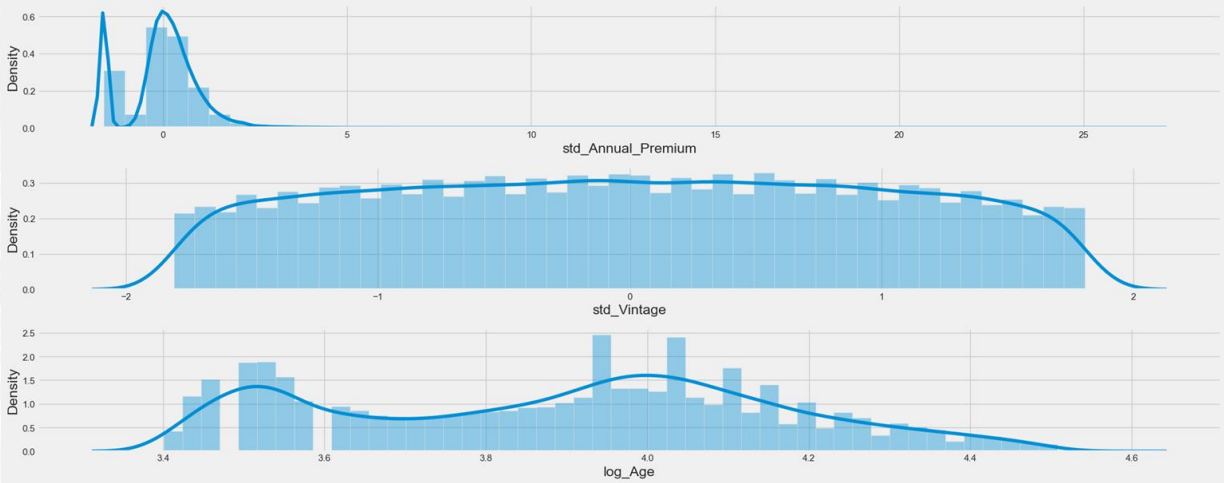
Data Preprocessing

Feature Transformation

df_train

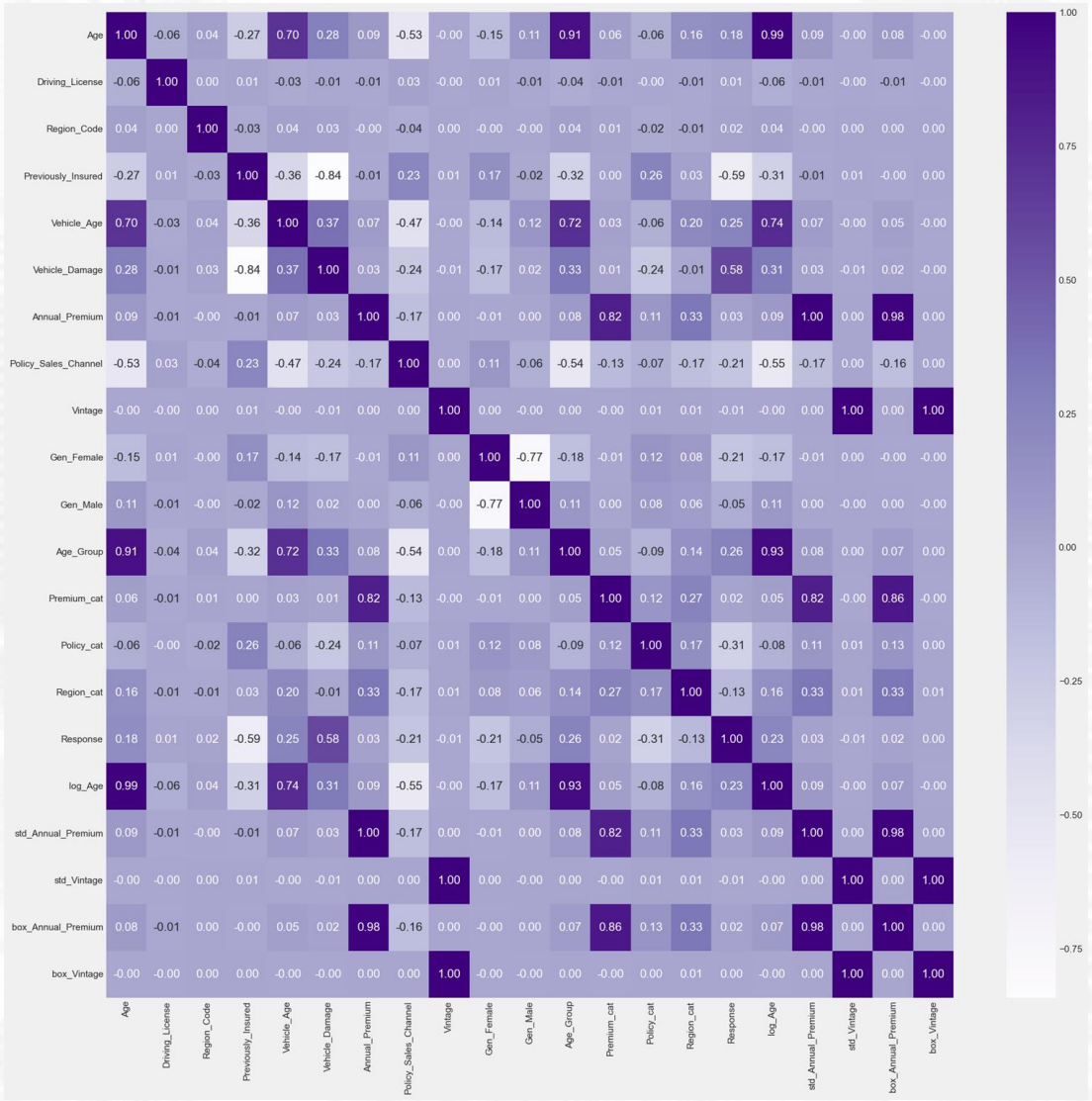


df_test



Data Preprocessing

Feature Transformation



Diputuskan menggunakan **StandardScaler**.

Features yang dipilih **Vehicle_Age**, **Vehicle_Damage**, **Previously_Insured**, **Gen_Female**, **Gen_Male**, **Age_Group**, **Region_cat**, **std_Annual_Premium**.

Sedangkan targetnya adalah **Response**.

Modelling

Pada modelling digunakan 7 algoritma klasifikasi yakni,

- **Logistic Regression**
- **K-Nearest Neighbor**
- **Decision Tree**
- **XGBoost**
- **Random Forest**
- **LightGBM**
- **Gradient Boost**

Pada pembuatan model pertama dengan features,

```
features = ['Vehicle_Age','Vehicle_Damage','Previously_Insured','Gen_Female','Gen_Male','Age_Group','Region_cat','std_Ann
target = ['Response']
```

didapatkan hasil bahwa model-model *overfitting*

Model	Accuracy Test	Accuracy Train	Precision Test	Precision Train	Recall Test	Recall Train	F1 Test	F1 Train	ROC AUC Test	ROC AUC Train	ROC AUC CrossVal Test	ROC AUC CrossVal Train
Logistic	0.79	0.78	0.71	0.71	0.98	0.98	0.82	0.82	0.82	0.82	0.99	0.80
KNN	0.76	0.81	0.72	0.77	0.82	0.88	0.77	0.82	0.81	0.89	0.99	0.80
Decision Tree	0.72	0.95	0.71	0.93	0.75	0.97	0.73	0.95	0.73	0.99	0.99	0.80
XGBoost	0.79	0.80	0.72	0.73	0.93	0.94	0.81	0.82	0.83	0.87	0.99	0.80
Random Forest	0.73	0.95	0.71	0.92	0.76	0.98	0.73	0.95	0.81	0.99	0.99	0.80
LightGBM	0.79	0.79	0.72	0.73	0.93	0.94	0.81	0.82	0.84	0.85	0.99	0.80
Gradient Boost	0.79	0.79	0.72	0.72	0.93	0.94	0.82	0.82	0.84	0.84	0.99	0.80

Modelling

Sehingga diputuskan untuk melakukan regularization dan hyperparameter tuning lain namun model masih overfitting maka dipertimbangkan untuk melakukan features selection ulang dan penambahan data.

```
# New Feature Selection
features_new = ['Vehicle_Age','Vehicle_Damage','Previously_Insured','Age_Group','Region_cat','Policy_Sales_Channel','Gen_
target_new = ['Response']

X = df[features_new].copy()
y = df[target_new].copy()
X_train = df_train[features_new].copy()
y_train = df_train[target_new].copy()
X_test = df_test[features_new].copy()
y_test = df_test[target_new].copy()
```

Modelling

```
def eval_classification(model):
    y_pred = model.predict(X_test)
    y_pred_train = model.predict(X_train)
    y_pred_proba = model.predict_proba(X_test)
    y_pred_proba_train = model.predict_proba(X_train)

    print("Accuracy (Test Set): %.2f" % accuracy_score(y_test, y_pred))
    print("Accuracy (Train Set): %.2f" % accuracy_score(y_train, y_pred_train))
    print("Precision (Test Set): %.2f" % precision_score(y_test, y_pred))
    print("Precision (Train Set): %.2f" % precision_score(y_train, y_pred_train))
    print("Recall (Test Set): %.2f" % recall_score(y_test, y_pred))
    print("Recall (Train Set): %.2f" % recall_score(y_train, y_pred_train))
    print("F1-Score (Test Set): %.2f" % f1_score(y_test, y_pred))
    print("F1-Score (Train Set): %.2f" % f1_score(y_train, y_pred_train))

    print("roc_auc (test-proba): %.2f" % roc_auc_score(y_test, y_pred_proba[:, 1]))
    print("roc_auc (train-proba): %.2f" % roc_auc_score(y_train, y_pred_proba_train[:, 1]))

    score = cross_validate(RandomForestClassifier(), X, y, cv=5, scoring='roc_auc', return_train_score=True)
    print('roc_auc (crossval train): '+ str(score['train_score'].mean()))
    print('roc_auc (crossval test): '+ str(score['test_score'].mean()))

def show_feature_importance(model):
    feat_importances = pd.Series(model.feature_importances_, index=X.columns)
    ax = feat_importances.nlargest(25).plot(kind='barh', figsize=(10, 8))
    ax.invert_yaxis()

    plt.xlabel('score')
    plt.ylabel('feature')
    plt.title('feature importance score')

def show_best_hyperparameter(model):
    print(model.best_estimator_.get_params())

lg = LogisticRegression(random_state=42)
knn = KNeighborsClassifier()
dt = DecisionTreeClassifier(random_state=42)
xgb = XGBClassifier(random_state=42)
rf = RandomForestClassifier(random_state=42)
lgb = LGBMClassifier(random_state=42)
grd = GradientBoostingClassifier(random_state=42)
```

→ Membuat function yang akan digunakan selama proses modelling berlangsung.

Modelling

Pada pembuatan model dengan *features* baru menggunakan 7 algoritma klasifikasi yang sama yakni,

- **Logistic Regression**
- **K-Nearest Neighbor**
- **Decision Tree**
- **XGBoost**
- **Random Forest**
- **LightGBM**
- **Gradient Boost**

Dihasilkan evaluasi yang cukup baik,

Model	Accuracy Test	Accuracy Train	Precision Test	Precision Train	Recall Test	Recall Train	F1 Test	F1 Train	ROC AUC Test	ROC AUC Train	ROC AUC CrossVal Test	ROC AUC CrossVal Train
Logistic	0.79	0.79	0.72	0.72	0.93	0.94	0.82	0.82	0.87	0.87	0.93	0.92
KNN	0.79	0.80	0.78	0.78	0.83	0.83	0.80	0.80	0.89	0.89	0.93	0.92
Decision Tree	0.82	0.83	0.78	0.79	0.89	0.90	0.83	0.84	0.91	0.93	0.93	0.92
XGBoost	0.82	0.83	0.78	0.78	0.91	0.91	0.84	0.84	0.92	0.92	0.93	0.92
Random Forest	0.82	0.83	0.78	0.79	0.89	0.90	0.83	0.84	0.92	0.93	0.93	0.92
LightGBM	0.82	0.82	0.77	0.77	0.91	0.91	0.84	0.84	0.92	0.92	0.93	0.92
Gradient Boost	0.82	0.82	0.76	0.76	0.93	0.92	0.84	0.84	0.91	0.91	0.93	0.92

Dimana model yang akan dilakukan hyperparameter tuning adalah **Logistic Regression, Decision Tree, XGBoost, Random Forest, LightGBM, Gradient Boost**.

Modelling

Rakamin Academy

Logistic Regression Hyperparameter Tuning

```
parameter = {'C': [float(x) for x in np.linspace(0.0001, 0.05, 100)],'penalty':['l2','l1', 'elasticnet']}
lgmodel2 = GridSearchCV(lg, parameter, cv = 3, n_jobs = -1 , verbose = 1, scoring = 'recall')
lgmodel2.fit(X_train, y_train)
lgmodelbest = lgmodel2.best_estimator_
y_pred_lg2 = lgmodelbest.predict(X_test)
y_pred_lg2_prob= lgmodelbest.predict_proba(X_test)[:,1]
```

```
Fitting 3 folds for each of 300 candidates, totalling 900 fits
```

Decision Tree Hyperparameter Tuning

```
parameter = {
            'max_depth':[int(x) for x in np.linspace(1, 110, num = 30)],
            'min_samples_split':[2, 5, 10, 100],
            'max_features':['auto', 'sqrt'],
            'criterion':['gini','entropy'],
            'splitter':['best','random']}

dtmodel2 = GridSearchCV(dt, parameter, cv=3, n_jobs = -1 , verbose = 1, scoring='recall')
dtmodel2.fit(X_train, y_train)
dtmodelbest = dtmodel2.best_estimator_
y_pred_dt2 = dtmodelbest.predict(X_test)
y_pred_dt2_prob= dtmodelbest.predict_proba(X_test)[:,1]
```

```
Fitting 3 folds for each of 960 candidates, totalling 2880 fits
```

Modelling

XGBoost Hyperparameter Tuning

```
parameter = {
            'max_depth' : [int(x) for x in np.linspace(10, 110, num = 11)],
            'min_child_weight' : [int(x) for x in np.linspace(1, 20, num = 11)],
            'gamma' : [float(x) for x in np.linspace(0, 1, num = 11)],
            'tree_method' : ['auto', 'exact', 'approx', 'hist'],
            'colsample_bytree' : [float(x) for x in np.linspace(0, 1, num = 11)],
            'eta' : [float(x) for x in np.linspace(0, 1, num = 100)],
            'lambda' : [float(x) for x in np.linspace(0, 1, num = 11)],
            'alpha' : [float(x) for x in np.linspace(0, 1, num = 11)]
            }

xgmodel2 = RandomizedSearchCV(xgb, parameter, cv=3, n_jobs = -1 , verbose = 1, scoring='recall', random_state=42)
xgmodel2.fit(X_train,y_train)
xgmodelbest = xgmodel2.best_estimator_
y_pred_xg2 = xgmodelbest.predict(X_test)
y_pred_xg2_prob= xgmodelbest.predict_proba(X_test)[:,1]
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
```

LightGBM Hyperparameter Tuning

```
f2 = make_scorer(fbeta_score, beta =2)
parameter = {"n_estimators":[200,300,400],"max_depth":[3,5,9]}
lgbmodel2 = HalvingGridSearchCV(lgb, parameter, scoring = f2, cv=3, n_jobs = -1, verbose = 1)
lgbmodel2.fit(X_train,y_train)
lgbmodel2.best_params_, lgbmodel2.best_score_
lgbmodelbest = LGBMClassifier(**lgbmodel2.best_params_)
lgbmodelbest.fit(X_train, y_train)
y_pred_lgb2 = lgbmodelbest.predict(X_test)
y_pred_lgb2_prob = lgbmodelbest.predict_proba(X_test)[:,1]
```

Modelling

Random Forest Hyperparameter Tuning

```
n_estimators = [int(x) for x in np.linspace(1, 200, 50)]
criterion = ['gini', 'entropy']
max_depth = [int(x) for x in np.linspace(2, 100, 50)]
min_samples_split = [int(x) for x in np.linspace(2, 20, 10)]
min_samples_leaf = [int(x) for x in np.linspace(2, 20, 10)]
parameter = dict(n_estimators=n_estimators, criterion=criterion, max_depth=max_depth,
                 min_samples_split=min_samples_split, min_samples_leaf=min_samples_leaf)

rsmodel2 = RandomizedSearchCV(rf, parameter, cv=3, n_jobs = -1 , verbose = 1, scoring='recall', random_state=42)
rsmodel2.fit(X_train,y_train)
rsmodelbest = rsmodel2.best_estimator_
y_pred_rs2 = rsmodelbest.predict(X_test)
y_pred_rs2_prob= rsmodelbest.predict_proba(X_test)[:,1]
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
```

Gradient Boost Hyperparameter Tuning

```
parameter = {'learning_rate':[0.15,0.1,0.05,0.01,0.005,0.001], 'n_estimators':[100,250,500,750,1000,1250,1500,1750], 'max

grdmodel2 = RandomizedSearchCV(grd, parameter, scoring='recall', n_jobs=-1, cv=3, verbose=1)
grdmodel2.fit(X_train,y_train)
grdmodel2.best_params_, grdmodel2.best_score_
grdmodelbest = GradientBoostingClassifier(**grdmodel2.best_params_)
grdmodelbest.fit(X_train, y_train)
y_pred_grd2 = grdmodelbest.predict(X_test)
y_pred_grd2_prob = grdmodelbest.predict_proba(X_test)[:,1]
```

Modelling

Score dari Hyperparameter Tuning

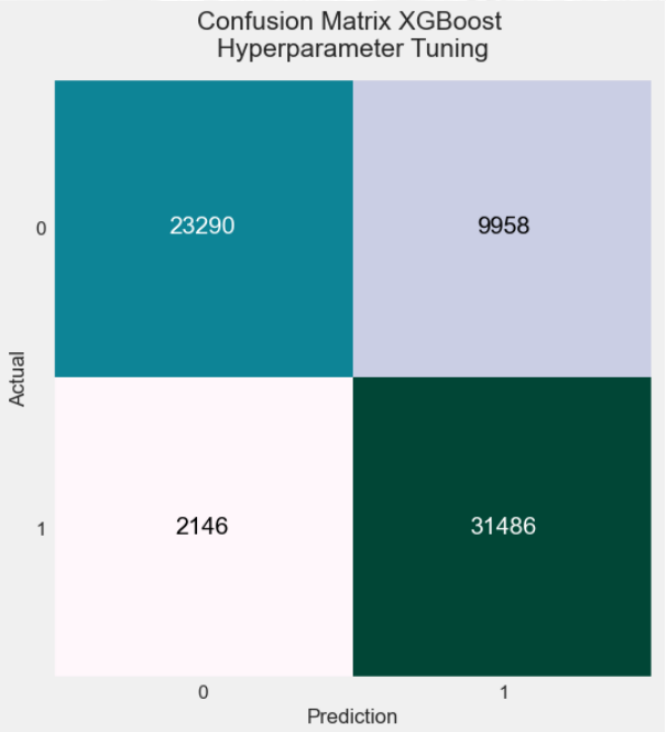
Model	Accuracy Test	Accuracy Train	Precision Test	Precision Train	Recall Test	Recall Train	F1 Test	F1 Train	ROC AUC Test	ROC AUC Train	ROC AUC CrossVal Test	ROC AUC CrossVal Train
Logistic	0.78	0.79	0.71	0.71	0.96	0.95	0.82	0.82	0.87	0.87	0.93	0.92
Decision Tree	0.79	0.79	0.73	0.72	0.94	0.94	0.82	0.82	0.86	0.86	0.93	0.92
XGBoost	0.82	0.82	0.76	0.76	0.94	0.94	0.84	0.84	0.92	0.92	0.93	0.92
Random Forest	0.81	0.81	0.74	0.74	0.95	0.95	0.83	0.83	0.89	0.89	0.93	0.92
LightGBM	0.82	0.82	0.77	0.77	0.92	0.92	0.84	0.84	0.92	0.92	0.93	0.92
Gradient Boost	0.82	0.82	0.76	0.76	0.93	0.93	0.84	0.83	0.91	0.91	0.93	0.92

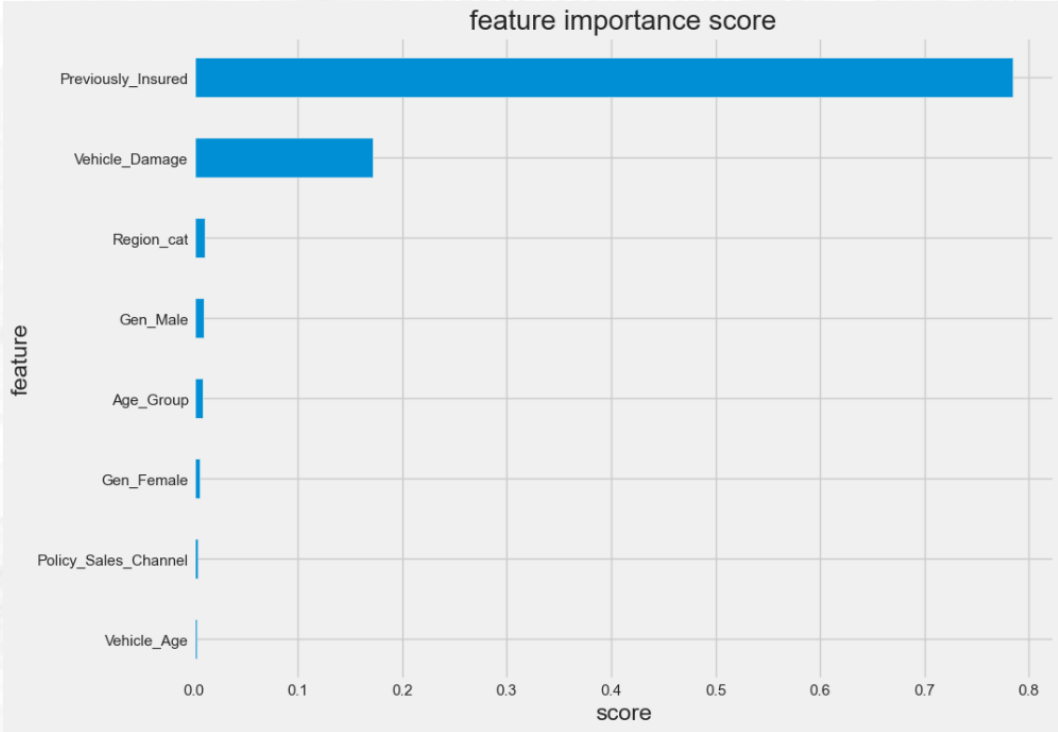
Modelling

The Best Fit Model

XGBoost Model

Model ini memiliki score recall yang tinggi yakni mencapai **0.94** dengan probabilitas machine learning sebesar **0.92** dan AUC ROC mencapai **0.91** pada hasil test data. Sedangkan pada train data nya hanya memiliki selisih terbesar 0.02 dari test data, yakni model tidak overfit maupun underfit yang dapat disebut sebagai ***model best fit***.

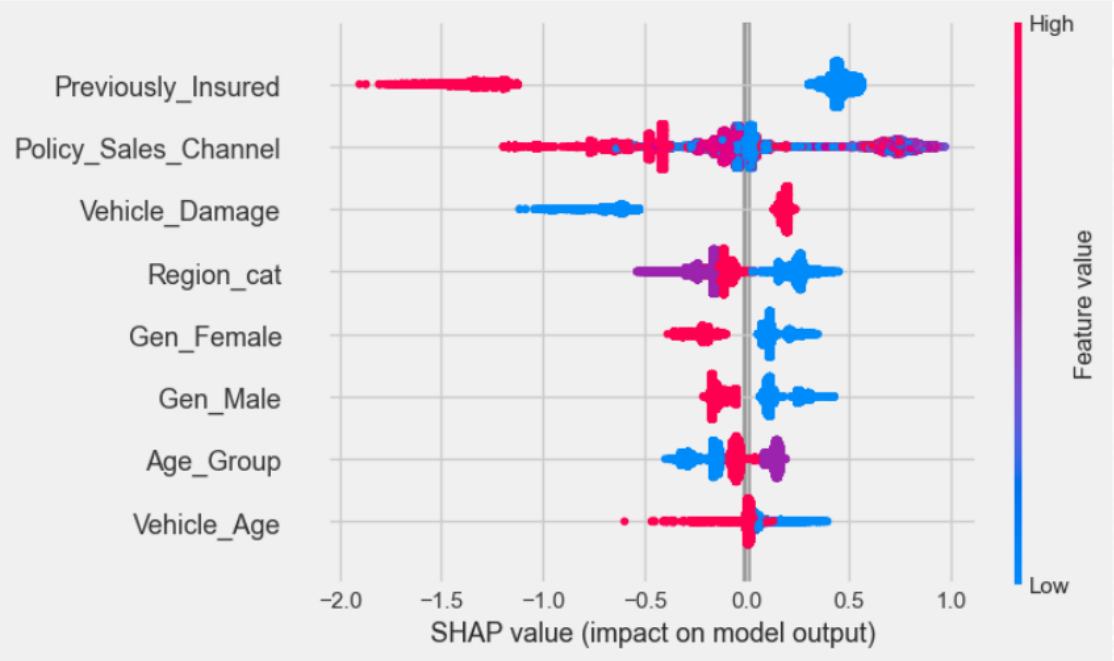




Pada model **XGBoost**, 2 features yang memiliki importance terbesar adalah **Previously_Insured** dan **Vehicle_Damage** sedangkan 2 features yang paling rendah adalah **Policy_Sales_Channel** dan **Vehicle_Age**.

Modelling

Shap Values Summary



- **'Previously_Insured'**: Value yang bernilai tinggi memiliki kontribusi negatif dengan hasil prediksi sedangkan value yang nilainya rendah memiliki kontribusi positif terhadap hasil prediksi.
- **'Policy_Sales_Channel'**: Semakin tinggi value berdampak kontribusi negatif terhadap prediksi sedangkan semakin rendah value memiliki kontribusi positif terhadap hasil prediksi.
- **'Vehicle_Damage'**: Semakin value bernilai tinggi memiliki kontribusi positif terhadap prediksi dan semakin rendah nilai value memiliki kontribusi negatif terhadap prediksi.
- **'Region_cat'**: Value yang bernilai tengah ke tinggi memiliki kontribusi negatif terhadap prediksi dan sebaliknya value yang bernilai rendah memiliki kontribusi yang positif.
- **'Gen_Female'**: Value tinggi berkontribusi negatif dan value rendah berkontribusi positif.
- **'Gen_Male'**: Value tinggi berkontribusi negatif dan value rendah berkontribusi positif.
- **'Age_Group'**: Value yang bernilai rendah dan tinggi memiliki kontibusi yang negatif terhadap prediksi sedangkan value yang memiliki nilai ditengah-tengah memiliki kontribusi yang positif.
- **'Vehicle_Age'**: Semakin tinggi value semakin negatif kontribusinya, dan semakin rendah value semakin positif kontribusinya terhadap prediksi.

Modelling

Business Recommendation

- Dari insight diatas dapat disimpulkan bahwa belum atau sudahnya seseorang dalam memiliki asuransi menjadi hal yang sangat berpengaruh sehingga penargetan marketing kepada customers yang belum memiliki asuransi sangat direkomendasikan.
- Jenis kelamin tidak memiliki pengaruh yang signifikan terhadap minat atau tidaknya seseorang terhadap asuransi kendaraan.
- Group umur yang paling memiliki ketertarikan tinggi terhadap asuransi adalah Group umur tengah-tengah yang disimpulkan adalah *YoungAdults*.
- Kebanyakan customers yang memiliki kendaraan rusak (damaged) akan cenderung lebih menunjukan ketertarikan terhadap asuransi kendaraan.
- Fokus pemasaran ada code region yang rendah.

Modelling

```
import pickle
pickle.dump(xgmodelbest, open('XGBoost_Model.pkl', 'wb'))
```

menyimpan (serialize) objek Python ke dalam
format biner dan menyimpannya dalam sebuah file.

5. Git

A. Buat Repository Git

B. Upload file notebook atau file pengerjaan lainnya pada repository tersebut

Click on Image

Alternatif **Link**.


Rakamin
Academy
THANK YOU