Rakamin Academy

Group 1

Dokumen
Laporan Final Project

Stage 3

1. Modeling

A. Split Data Train & Test

Pada proses modeling di stage 3 ini, kami menggunakan beberapa scenario data untuk menganalisa pengaruh dari setiap feature terhadap performa model. Berikut adalah beberapa scenario data yang kami gunakan beserta dengan split data train & test nya

```
df1 = df[['Age','Employment Type','AnnualIncome','FrequentFlyer','EverTravelledAbroad','TravelInsurance']]
df2 = df[['Age','Employment Type','AnnualIncome','Traveller','TravelInsurance']]
df3 = df[['Age_Bracket','Employment Type','AnnualIncome','Traveller','TravelInsurance']]
df4 = df[['Age_Bracket','Employment Type','Income_Bracket','Traveller','TravelInsurance']]
df5 = df[['Age','Employment Type','Income_Bracket','Traveller','TravelInsurance']]
df6 = df[['Age','Employment Type','Income_Bracket','FrequentFlyer','EverTravelledAbroad','TravelInsurance']]
```

- df1 merupakan keseluruhan feature selection yaitu (Age, Employment Type, Annual Income, Frequent Flyer, Ever Travelled Abroad, dan Travel Insurance).
- df2 menggunakan feature extraction Traveller yang merupakan gabungan dari feature Frequent Flyer dan Ever Travelled Abroad. Fungsinya adalah untuk melihat impact feature Traveller terhadap performa model.
- df3 sama seperti df2, akan tetapi untuk feature Age, diubah menjadi feature Age_Bracket. Tujuannya untuk melihat impact dari Age_Bracket terhadap performa model dari df2.
- df4 sama seperti df3, akan tetapi untuk feature Annual Income, diubah menjadi feature Income_Bracket. Tujuannya untuk melihat impact dari Income_Bracket terhadap performa model dari df3.
- df5 sama seperti df4, akan tetapi untuk feature Age_Brakcket, diubah menjadi feature Age. Tujuannya, untuk melihat perubahan performa model dari df4 jika tanpa feature Age_Bracket.
- sama seperti df1, akan tetapi untuk feature Traveller, diubah kembali menjadi feature Frequent Flyer dan Ever Travelled Abroad. Tujuannya untuk melihat perubahan performa model jika hanya ada perubahan feature Annual Income diubah menjadi Income Bracket.

1. Modeling

A. Split Data Train & Test (1/6)

Berikut adalah hasil split data train & test untuk **df1**

```
X = df1.drop(columns=['TravelInsurance'])
y = df1['TravelInsurance']

df1.shape
```

```
(1249, 6)
```

```
#Split data train & test df1
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
X_train.shape
```

```
(874, 5)
```

```
X_test.shape
```

```
(375, 5)
```

1. Modeling

A. Split Data Train & Test (2/6)

Berikut adalah hasil split data train & test untuk **df2**

```
X = df2.drop(columns=['TravelInsurance'])
y = df2['TravelInsurance']

df2.shape
```

```
(1249, 5)
```

```
#Split data train & test df1
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
X_train.shape
```

```
(874, 4)
```

```
X_test.shape
```

```
(375, 4)
```

1. Modeling

A. Split Data Train & Test (3/6)

Berikut adalah hasil split data train & test untuk **df3**

```
X = df3.drop(columns=['TravelInsurance'])
y = df3['TravelInsurance']

df3.shape
```

```
(1249, 5)
```

```
#Split data train & test df1
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
X_train.shape
```

```
(874, 4)
```

```
X_test.shape
```

```
(375, 4)
```

1. Modeling

A. Split Data Train & Test (4/6)

Berikut adalah hasil split data train & test untuk **df4**

```
X = df4.drop(columns=['TravelInsurance'])
y = df4['TravelInsurance']

df4.shape
```

```
(1249, 5)
```

```
#Split data train & test df1
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
X_train.shape
```

```
(874, 4)
```

```
X_test.shape
```

```
(375, 4)
```

1. Modeling

A. Split Data Train & Test (5/6)

Berikut adalah hasil split data train & test untuk **df5**

```
X = df5.drop(columns=['TravelInsurance'])
y = df5['TravelInsurance']

df5.shape
```

```
(1249, 5)
```

```
#Split data train & test df1
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
X_train.shape
```

```
(874, 4)
```

```
X_test.shape
```

```
(375, 4)
```

1. Modeling

A. Split Data Train & Test (6/6)

Berikut adalah hasil split data train & test untuk **df6**

```
X = df6.drop(columns=['TravelInsurance'])
y = df6['TravelInsurance']

df6.shape
```

```
(1249, 6)
```

```
#Split data train & test df1
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
X_train.shape
```

```
(874, 5)
```

```
X_test.shape
```

```
(375, 5)
```

1. Modeling

B. Modeling & Model Evaluation

Pada tahap modeling akan dilakukan percobaan dengan menggunakan algoritma berikut ini:

- Logistic Regression
- K-Nearest Neighbor
- Decision Tree
- Random Forest
- Adaboost

dengan menggunakan beberapa data scenario yang sudah dibuat untuk mencari performa terbaik model dilihat dari metrics model berikut ini:

- Accuracy
- **Precision**
- Recall
- F1-Score
- ROC-AUC

Dari lima metrics model tersebut, kelompok kami memilih untuk melihat performa model berdasarkan metrics model **precision** untuk memperhatikan jumlah False Positive (FP) yang sebaiknya lebih sedikit yaitu memperkecil kemungkinan munculnya prediksi customer membeli insurance travel sedangkan data actual menunjukkan customer tidak membeli insurance travel. Hal ini dilakukan untuk memperkecil cost marketing.

1. Modeling

B. Modeling & Model Evaluation – Logistic Regression

Dengan menggunakan fungsi berikut

```
logreg = LogisticRegression(random_state = 42) # inisiasi object dengan nama logreg
logreg.fit(X_train, y_train) # fit model regression dari data train
eval_classification(logreg)
```

diperoleh hasil sebagai berikut

df1

```
Accuracy (Test Set): 0.62
Precision (Train Set): 0.00
Precision (Test Set): 0.00
Recall (Test Set): 0.00
F1-Score (Test Set): 0.00
ROC-AUC (Test-proba): 0.31
```

df2

```
Accuracy (Test Set): 0.62
Precision (Train Set): 0.00
Precision (Test Set): 0.00
Recall (Test Set): 0.00
F1-Score (Test Set): 0.00
ROC-AUC (Test-proba): 0.31
```

df3

```
Accuracy (Test Set): 0.62
Precision (Train Set): 0.00
Precision (Test Set): 0.00
Recall (Test Set): 0.00
F1-Score (Test Set): 0.00
ROC-AUC (Test-proba): 0.31
```

df4

```
Accuracy (Test Set): 0.74
Precision (Train Set): 0.84
Precision (Test Set): 0.87
Recall (Test Set): 0.38
F1-Score (Test Set): 0.53
ROC-AUC (Test-proba): 0.69
```

df5

```
Accuracy (Test Set): 0.74
Precision (Train Set): 0.84
Precision (Test Set): 0.87
Recall (Test Set): 0.38
F1-Score (Test Set): 0.53
ROC-AUC (Test-proba): 0.69
```

df6

```
Accuracy (Test Set): 0.74
Precision (Train Set): 0.80
Precision (Test Set): 0.85
Recall (Test Set): 0.40
F1-Score (Test Set): 0.54
ROC-AUC (Test-proba): 0.68
```

1. Modeling

B. Modeling & Model Evaluation – Logistic Regression (2/2)

Kesimpulan yang diperoleh

1. Perfoma model untuk data scenario 1, 2, dan 3 menunjukkan angka 0 karena??
2. Pada saat menggunakan data scenario 4, performa model langsung mengalami peningkatan ke angka 87% . Artinya, feature Income Bracket memiliki feature importance yang tinggi pada performa model.
3. Berdasarkan performa model di point 2, pada data scenario 5 dan 6 dilakukan percobaan untuk melihat performa model dengan mengubah feature Age Bracket menjadi Age kembali begitu juga dengan feature Traveller diubah kembali menjadi Frequent Flyer dan Ever Travelled Abroad. Hasilnya tidak ada perubahan performa model untuk data scenario 5 dan terjadi penurunan 2% dengan menggunakan data scenario 6.
4. Dari penjelasan 3 point di atas, dapat disimpulkan bahwa feature Income Bracket sangat berpengaruh pada performa model.

1. Modeling

B. Modeling & Model Evaluation – K-Nearest Neighbor (1/2)

Dengan menggunakan fungsi berikut

```
knn = KNeighborsClassifier() # inisiasi object dengan nama knn
knn.fit(X_train, y_train) # fit model KNN dari data train
eval_classification(knn)
```

diperoleh hasil sebagai berikut

df1

```
Accuracy (Test Set): 0.65
Precision (Train Set): 0.73
Precision (Test Set): 0.56
Recall (Test Set): 0.44
F1-Score (Test Set): 0.49
ROC-AUC (Test-proba): 0.66
```

df2

```
Accuracy (Test Set): 0.66
Precision (Train Set): 0.70
Precision (Test Set): 0.57
Recall (Test Set): 0.45
F1-Score (Test Set): 0.50
ROC-AUC (Test-proba): 0.67
```

df3

```
Accuracy (Test Set): 0.72
Precision (Train Set): 0.71
Precision (Test Set): 0.70
Recall (Test Set): 0.48
F1-Score (Test Set): 0.57
ROC-AUC (Test-proba): 0.70
```

df4

```
Accuracy (Test Set): 0.62
Precision (Train Set): 0.51
Precision (Test Set): 0.50
Recall (Test Set): 0.65
F1-Score (Test Set): 0.56
ROC-AUC (Test-proba): 0.69
```

df5

```
Accuracy (Test Set): 0.74
Precision (Train Set): 0.77
Precision (Test Set): 0.81
Recall (Test Set): 0.42
F1-Score (Test Set): 0.55
ROC-AUC (Test-proba): 0.67
```

df6

```
Accuracy (Test Set): 0.72
Precision (Train Set): 0.77
Precision (Test Set): 0.71
Recall (Test Set): 0.44
F1-Score (Test Set): 0.55
ROC-AUC (Test-proba): 0.69
```

1. Modeling

B. Modeling & Model Evaluation – K-Nearest Neighbor (2/2)

Kesimpulan yang diperoleh

1. Perfoma model dengan menggunakan algoritma KNN memiliki hasil yang cukup beragam pada setiap data scenario yang digunakan. Hal ini terjadi kemungkinan besar karena belum dilakukan feature transformation pada data yang dipakai . Sedangkan algoritma ini memperhitungkan jarak pada prosesnya.
2. Selanjutnya perlu dilakukan feature transformation jika memakai algoritma KNN ini.

1. Modeling

B. Modeling & Model Evaluation – Decision Tree (1/2)

Dengan menggunakan fungsi berikut

```
dt = DecisionTreeClassifier(random_state=42) # inisiasi object dengan nama dt
dt.fit(X_train, y_train) # fit model decision tree dari data train
eval_classification(dt)
```

diperoleh hasil sebagai berikut

df1

```
Accuracy (Test Set): 0.65
Precision (Train Set): 0.81
Precision (Test Set): 0.55
Recall (Test Set): 0.40
F1-Score (Test Set): 0.47
ROC-AUC (Test-proba): 0.64
```

df2

```
Accuracy (Test Set): 0.67
Precision (Train Set): 0.77
Precision (Test Set): 0.59
Recall (Test Set): 0.44
F1-Score (Test Set): 0.51
ROC-AUC (Test-proba): 0.68
```

df3

```
Accuracy (Test Set): 0.71
Precision (Train Set): 0.79
Precision (Test Set): 0.72
Recall (Test Set): 0.41
F1-Score (Test Set): 0.52
ROC-AUC (Test-proba): 0.69
```

df4

```
Accuracy (Test Set): 0.74
Precision (Train Set): 0.85
Precision (Test Set): 0.89
Recall (Test Set): 0.38
F1-Score (Test Set): 0.53
ROC-AUC (Test-proba): 0.69
```

df5

```
Accuracy (Test Set): 0.74
Precision (Train Set): 0.78
Precision (Test Set): 0.78
Recall (Test Set): 0.43
F1-Score (Test Set): 0.56
ROC-AUC (Test-proba): 0.70
```

df6

```
Accuracy (Test Set): 0.73
Precision (Train Set): 0.84
Precision (Test Set): 0.78
Recall (Test Set): 0.42
F1-Score (Test Set): 0.55
ROC-AUC (Test-proba): 0.70
```

1. Modeling

B. Modeling & Model Evaluation – Random Forest (1/2)

Dengan menggunakan fungsi berikut

```
rf = RandomForestClassifier(random_state=42) # inisiasi object dengan nama rf
rf.fit(X_train, y_train) # fit model random forest dari data train
eval_classification(rf)
```

diperoleh hasil sebagai berikut

df1

```
Accuracy (Test Set): 0.65
Precision (Train Set): 0.78
Precision (Test Set): 0.55
Recall (Test Set): 0.45
F1-Score (Test Set): 0.50
ROC-AUC (Test-proba): 0.67
```

df2

```
Accuracy (Test Set): 0.65
Precision (Train Set): 0.74
Precision (Test Set): 0.56
Recall (Test Set): 0.47
F1-Score (Test Set): 0.51
ROC-AUC (Test-proba): 0.68
```

df3

```
Accuracy (Test Set): 0.71
Precision (Train Set): 0.76
Precision (Test Set): 0.70
Recall (Test Set): 0.43
F1-Score (Test Set): 0.53
ROC-AUC (Test-proba): 0.69
```

df4

```
Accuracy (Test Set): 0.74
Precision (Train Set): 0.85
Precision (Test Set): 0.89
Recall (Test Set): 0.38
F1-Score (Test Set): 0.53
ROC-AUC (Test-proba): 0.69
```

df5

```
Accuracy (Test Set): 0.74
Precision (Train Set): 0.78
Precision (Test Set): 0.78
Recall (Test Set): 0.43
F1-Score (Test Set): 0.56
ROC-AUC (Test-proba): 0.70
```

df6

```
Accuracy (Test Set): 0.73
Precision (Train Set): 0.83
Precision (Test Set): 0.75
Recall (Test Set): 0.42
F1-Score (Test Set): 0.54
ROC-AUC (Test-proba): 0.71
```

1. Modeling

B. Modeling & Model Evaluation – Random Forest (2/2)

Kesimpulan yang diperoleh

1. Performa model dengan menggunakan algoritma Decision Tree mirip dengan algoritma Random Forest untuk setiap data scenario karena algoritma Random Forest merupakan turunan dari algoritma Decision Tree.
2. Hasil model dengan menggunakan data scenario 1, 2, 3 masih mengalami overfitting.
3. Hasil model dengan menggunakan data scenario 2 tidak ada perbedaan signifikan dengan performa model dengan menggunakan data scenario 1. Artinya, feature extraction Frequent Flyer dan Ever Travelled Abroad mejadi Traveller tidak memberikan pengaruh apapun pada performa model.
4. Dengan menggunakan data scenario 3, performa model naik 24%. Hal ini berarti, adanya feature Age Bracket berpengaruh pada performa model.
5. Performa model terbaik ditunjukkan saat menggunakan data scenario 4, dimana precision mencapai angka 89%. Artinya, perubahan feature Annual Income menjadi Income Bracket, berpengaruh pada performa model.

1. Modeling

B. Modeling & Model Evaluation – Adaboost (1/2)

Dengan menggunakan fungsi berikut

```
clf = AdaBoostClassifier(random_state=42) # inisiasi object dengan nama clf
clf.fit(X_train, y_train) # fit model Adaboost dari data train
eval_classification(clf)
```

diperoleh hasil sebagai berikut

df1

```
Accuracy (Test Set): 0.74
Precision (Train Set): 0.75
Precision (Test Set): 0.82
Recall (Test Set): 0.43
F1-Score (Test Set): 0.56
ROC-AUC (Test-proba): 0.72
```

df2

```
Accuracy (Test Set): 0.75
Precision (Train Set): 0.80
Precision (Test Set): 0.84
Recall (Test Set): 0.42
F1-Score (Test Set): 0.56
ROC-AUC (Test-proba): 0.72
```

df3

```
Accuracy (Test Set): 0.75
Precision (Train Set): 0.89
Precision (Test Set): 0.92
Recall (Test Set): 0.38
F1-Score (Test Set): 0.53
ROC-AUC (Test-proba): 0.70
```

df4

```
Accuracy (Test Set): 0.74
Precision (Train Set): 0.84
Precision (Test Set): 0.87
Recall (Test Set): 0.38
F1-Score (Test Set): 0.53
ROC-AUC (Test-proba): 0.69
```

df5

```
Accuracy (Test Set): 0.74
Precision (Train Set): 0.78
Precision (Test Set): 0.82
Recall (Test Set): 0.41
F1-Score (Test Set): 0.55
ROC-AUC (Test-proba): 0.72
```

df6

```
Accuracy (Test Set): 0.74
Precision (Train Set): 0.76
Precision (Test Set): 0.80
Recall (Test Set): 0.42
F1-Score (Test Set): 0.55
ROC-AUC (Test-proba): 0.71
```

1. Modeling

B. Modeling & Model Evaluation – Adaboost (2/2)

Kesimpulan yang diperoleh

1. Modeling dengan menggunakan algoritma Adaboost menunjukkan performa terbaik 92% pada saat menggunakan data scenario 3 yaitu dengan adanya feature extraction Age Bracket dan Traveller.
2. Sedangkan untuk model dengan data scenario 4, menunjukkan performa turun ke 87%.
3. Untuk analisa lebih dalam, akan dilakukan cek dengan menggunakan k-fold cross validation di section selanjutnya.

1. Modeling

B. Modeling & Model Evaluation Result

Berdasarkan hasil modeling dan model evaluation yang sudah dilakukan, dipilih model dengan menggunakan algoritma **Decision Tree** dengan metrics model **Precision** 89% pada data scenario 4 yang terdiri dari feature **'Age Bracket','Employment Type','Income Bracket', dan 'Traveller'**

```
Accuracy (Test Set): 0.74
Precision (Train Set): 0.85
Precision (Test Set): 0.89
Recall (Test Set): 0.38
F1-Score (Test Set): 0.53
ROC-AUC (Test-proba): 0.69
```

1. Modeling

D. Model Evaluation by Cross Validation – Logistic Regresion (1/2)

Dengan menggunakan fungsi berikut:

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

Diperoleh hasil berikut:

df1	df2	df3
Precision (crossval train): 0.0 Precision (crossval test): 0.0	Precision (crossval train): 0.0 Precision (crossval test): 0.0	Precision (crossval train): 0.0 Precision (crossval test): 0.0

df4	df5	df6
Precision (crossval train): 0.84728 Precision (crossval test): 0.843102	Precision (crossval train): 0.84865 Precision (crossval test): 0.843102	Precision (crossval train): 0.84470 Precision (crossval test): 0.843392

1. Modeling

D. Model Evaluation by Cross Validation – K-Nearest Neighbor(1/2)

Dengan menggunakan fungsi berikut:

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

Diperoleh hasil berikut:

df1	df2	df3
Precision (crossval train): 0.75309 Precision (crossval test): 0.615057	Precision (crossval train): 0.71115 Precision (crossval test): 0.634130	Precision (crossval train): 0.68009 Precision (crossval test): 0.616525

df4	df5	df6
Precision (crossval train): 0.85992 Precision (crossval test): 0.847693	Precision (crossval train): 0.65213 Precision (crossval test): 0.640382	Precision (crossval train): 0.68176 Precision (crossval test): 0.670678

1. Modeling

D. Model Evaluation by Cross Validation – Decision Tree (1/2)

Dengan menggunakan fungsi berikut:

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

Diperoleh hasil berikut:

df1	df2	df3
Precision (crossval train): 0.84926 Precision (crossval test): 0.652653	Precision (crossval train): 0.81593 Precision (crossval test): 0.684924	Precision (crossval train): 0.81107 Precision (crossval test): 0.734819

df4	df5	df6
Precision (crossval train): 0.86287 Precision (crossval test): 0.851000	Precision (crossval train): 0.8078 Precision (crossval test): 0.75957	Precision (crossval train): 0.85536 Precision (crossval test): 0.786042

1. Modeling

D. Model Evaluation by Cross Validation – Random Forest (1/2)

Dengan menggunakan fungsi berikut:

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

Diperoleh hasil berikut:

df1	df2	df3
Precision (crossval train): 0.79449 Precision (crossval test): 0.598028	Precision (crossval train): 0.78282 Precision (crossval test): 0.647539	Precision (crossval train): 0.78299 Precision (crossval test): 0.712561

df4	df5	df6
Precision (crossval train): 0.86005 Precision (crossval test): 0.833398	Precision (crossval train): 0.80340 Precision (crossval test): 0.759570	Precision (crossval train): 0.84275 Precision (crossval test): 0.787895

1. Modeling

D. Model Evaluation by Cross Validation – Adaboost (1/2)

Dengan menggunakan fungsi berikut:

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

Diperoleh hasil berikut:

df1	df2	df3
Precision (crossval train): 0.7972 Precision (crossval test): 0.77897	Precision (crossval train): 0.81062 Precision (crossval test): 0.796733	Precision (crossval train): 0.89919 Precision (crossval test): 0.895679

df4	df5	df6
Precision (crossval train): 0.85005 Precision (crossval test): 0.843102	Precision (crossval train): 0.79861 Precision (crossval test): 0.791209	Precision (crossval train): 0.78587 Precision (crossval test): 0.771639

1. Modeling

D. Model Evaluation by Cross Validation

Dari hasil k-fold cross validation beberapa algoritma machine learning yang sudah dilakukan, bisa diketahui bahwa untuk model **Decision Tree** dengan **data scenario 4** sudah best fit di mana crossval train lebih besar dari crossval test dengan selisih yang tidak terlalu jauh. Oleh karena itu tidak dilakukan hyperparameter tuning di tahap selanjutnya.

```
Precision (crossval train): 0.86287
Precision (crossval test): 0.851000
```

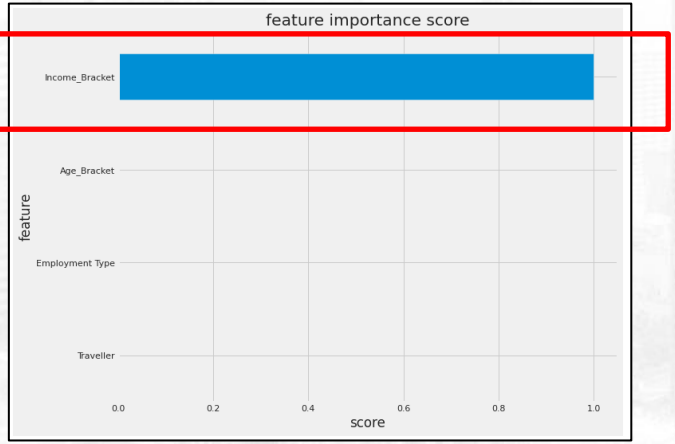
1. Modeling

E. Hyperparameter Tuning

Pada stage ini, tidak dilakukan hyperparameter tuning sesuai dengan penjelasan pada section sebelumnya bahwa model sudah best fit.

2. Feature Importance

Tahap selanjutnya adalah melihat feature importance. Di antara feature age bracket, employment type, income bracket, dan traveller diperoleh hasil sebagai berikut




```
X_train.columns

Index(['Age_Bracket', 'Employment Type', 'Income_Bracket', 'Traveller'], dtype='object')

dt.feature_importances_

array([0.07972106, 0.01378198, 0.88200841, 0.02448855])
```

- Berdasarkan hasil yang diperlihatkan oleh feature importance didapatkan informasi bahwa fitur yang memberikan pengaruh paling besar kepada model adalah fitur income bracket.
- Business insight yang dapat diperoleh adalah hal yang paling mempengaruhi apakah seorang customer akan membeli atau tidak sebuah travel insurance adalah tergantung dengan pendapatan mereka.
- Berdasarkan kedua poin di atas, pihak perusahaan atau marketing bisa mempertimbangkan agar harga dari travel insurance bisa dibuat menyesuaikan dengan mayoritas dari customer, atau bisa juga dengan membuat paket bundling untuk tiket perjalanan yang dibeli oleh customer, atau bisa juga dengan melakukan “penyamaran” untuk travel insurance langsung ke dalam pembelian tiket sebagai “bonus” atau “hadiah”