

29.0 Out[50]: 15.0 31495 14.0 24027 13.0 23476 5.0 21216 10.0 20454 16.0 17873 17.0 17603 11.0 17026 4.0 15991 6.0 15443 14909 9.0 18.0 14065 22.0 11532 21.0 10823 12.0 9229 23.0 9150 27.0 8475 7.0 8022 20.0 7383 26.0 7319 28.0 7167 8.0 7075 19.0 5336 25.0 5273 24.0 3138 Name: accountAgeMonth, dtype: int64 In [51]: df2.info() <class 'pandas.core.frame.DataFrame'> RangeIndex: 481092 entries, 0 to 481091 Data columns (total 23 columns): Non-Null Count # Column _____ orderItemID 0 481092 non-null int64 orderDate 481092 non-null datetime64[ns] orderYear 481092 non-null float64 orderMonth 481092 non-null float64 481092 non-null float64 orderDated 5 481092 non-null object size deliveryDate 481092 non-null datetime64[ns] 7 481092 non-null float64 deliveryYear 8 deliveryMonth 481092 non-null float64 481092 non-null float64 9 deliveryDated 10 deliveryDuration 481092 non-null float64 11 itemID 481092 non-null int64 12 color 481092 non-null object 481092 non-null int64 13 manufacturerID 14 price 481092 non-null float64 15 customerID 481092 non-null int64 16 salutation 481092 non-null object 17 dateOfBirth 432200 non-null datetime64[ns] 18 state 481092 non-null object 19 customerAge 481092 non-null float64 20 creationDate 481092 non-null datetime64[ns] 21 accountAgeMonth 481092 non-null float64 22 returnShipment 481092 non-null int64 dtypes: datetime64[ns](4), float64(10), int64(5), object(4) memory usage: 84.4+ MB In [52]: #proses drop orderDate, deliveryDate, dateOfBirth, dan creationDate karna sudah direpresentasikan dengan attrib df2 = df2.drop(['orderDate', 'deliveryDate', 'dateOfBirth', 'creationDate'],axis=1) df2.head(10) In [53]: Out[53]: orderItemID orderYear orderMonth orderDated size deliveryYear deliveryMonth deliveryDated deliveryDuration itemID color manu 2012.0 0 2012.0 4.0 1.0 4.0 3.0 2.0 186 blue 1 М 1 2012.0 2.0 4.0 1.0 2012.0 4.0 3.0 71 brown 2 2012.0 4.0 1.0 Μ 2012.0 4.0 3.0 2.0 71 yellow 2012.0 4.0 2.0 2012.0 5.0 3.0 22 green 5 4 2012.0 2.0 S 2012.0 4.0 5.0 3.0 4.0 151 black 5 6 2012.0 XXL 2012.0 5.0 3.0 4.0 2.0 4.0 598 brown 7 S 6 2012.0 4.0 2.0 2012.0 4.0 5.0 3.0 15 black 7 8 2012.0 2.0 XXL 2012.0 3.0 1.0 32 brown 4.0 4.0 8 9 2012.0 4.0 2.0 XXL 2012.0 4.0 3.0 1.0 32 red 9 10 3.0 2012.0 4.0 2.0 XXL 2012.0 4.0 1.0 57 green In [54]: #melihat korelasi atribut returnShipment dengan atribut lainnya yang bertipe numerik corr pearson = df2.corr() plt.figure(figsize=(12,8)) sns.heatmap(corr pearson, cmap="RdBu r",annot=True) plt.title('Correlation between Numeric Variables') plt.show() Correlation between Numeric Variables 1.0 -0.31 0.029 -0.32 -0.032 0.19 0.1 0.061 -0.011 -0.28 orderItemID 0.82 0.82 -0.79 -0.05 -0.72 -0.024 0.19 0.016 -0.014 orderYear 0.4 -0.0023 -0.24 - 0.8 orderMonth -0.31 -0.79 0.022 -0.7 0.87 -0.027 -0.12 **-0.25** 0.084 0.091 -0.1 -0.0079 0.1 -0.038 - 0.6 orderDated - 0.029 -0.05 0.022 -0.0065 -0.78 -0.082 0.68 0.024 -0.016 0.4 -0.0027 -0.24 0.041 deliveryYear - 0.4 deliveryMonth -0.32 -0.72 0.87 0.0014 -0.780.012 0.025 -0.23 0.066 0.1 -0.099-0.0076 0.096 -0.036 deliveryDated --0.032 -0.024 -0.027 0.34 -0.082 0.012 -0.029-0.00960.0063 0.018 -0.0160.000240.0093-0.0036 - 0.2 deliveryDuration - 0.19 0.19 -0.12 -0.014 0.2 0.025 -0.029 0.23 -0.023 0.034 0.11 0.0018 -0.062 0.028 - 0.0 -0.25 -0.02 -0.23 -0.0096 0.23 0.18 0.44 -0.016 -0.21 0.026 manufacturerID - 0.1 0.016 0.0840.00073 0.024 0.066 0.0063 -0.023 0.17 0.065 0.072 -0.044 -0.027 0.02 -0.2price - 0.061 -0.014 0.091 -0.014 -0.016 0.1 0.018 0.034 0.18 0.065 0.054 -0.024 -0.019 0.13 0.44 0.072 0.054 -0.47 -0.0026 customerID -0.1 0.019 0.4 -0.099 -0.016 0.11 -0.077 -0.4-0.046 customerAge --0.011-0.00230.00790.00640.0027-0.00760.000240.0018 -0.016 -0.044 -0.024 -0.077 -0.6accountAgeMonth - -0.28 -0.24 0.1 0.0029 -0.24 0.096 0.0093 -0.062 -0.21 -0.027 -0.019 -0.47 0.019 0.13 -0.0026-0.046 0.019 itemID price manufacturerID customerAge accountAgeMonth returnShipment orderitemID orderYear orderMonth orderDated deliveryYea deliveryMonth deliveryDated deliveryDuration customerID df2["salutation"] = df2["salutation"].astype('category') In [55]: df2["state"] = df2["state"].astype('category') df2["color"] = df2["color"].astype('category') df2["size"] = df2["size"].astype('category') #untuk mengubah atribut kategorikal menjadi one hot method In [56]: df2 = pd.get dummies(data=df2, columns=['salutation','size','color','state']) #untuk melihat jumlah kolom setelah di one hot In [57]: df2.shape (481092, 56) Out[57]: In [58]: x = df2.drop("returnShipment", axis=1) y = df2.returnShipmentIn [59]: x.info() <class 'pandas.core.frame.DataFrame'> RangeIndex: 481092 entries, 0 to 481091 Data columns (total 55 columns): # Column Non-Null Count Dtype orderItemID 481092 non-null int64 1 orderYear 481092 non-null float64 481092 non-null float64 orderMonth 481092 non-null float64 orderDated 481092 non-null float64 deliveryYear 481092 non-null float64 deliveryMonth deliveryDated 481092 non-null float64 481092 non-null float64 deliveryDuration itemID 481092 non-null int64 8 manufacturerID 481092 non-null int64 9 481092 non-null float64 10 price 481092 non-null int64 11 customerID 12 customerAge 481092 non-null float64 481092 non-null float64 13 accountAgeMonth 481092 non-null uint8 14 salutation Mr 481092 non-null uint8 15 salutation Mrs 481092 non-null uint8 16 salutation Other 481092 non-null uint8 17 size L 481092 non-null uint8 18 size M 481092 non-null uint8 19 size S 481092 non-null uint8 20 size XL 481092 non-null uint8 21 size XS 481092 non-null uint8 22 size XXL 481092 non-null uint8 23 size XXS 481092 non-null uint8 24 size XXXL 481092 non-null uint8 25 size unsized 481092 non-null uint8 26 color black 481092 non-null uint8 27 color blue 481092 non-null uint8 28 color brown 481092 non-null uint8 29 color green 481092 non-null uint8 30 color grey 481092 non-null uint8 31 color orange 481092 non-null uint8 32 color other 481092 non-null uint8 33 color pink 481092 non-null uint8 34 color purple 481092 non-null uint8 481092 non-null uint8
481092 non-null uint8
481092 non-null uint8
481092 non-null uint8
481092 non-null uint8
481092 non-null uint8
481092 non-null uint8
481092 non-null uint8
481092 non-null uint8
481092 non-null uint8
481092 non-null uint8
481092 non-null uint8
481092 non-null uint8
481092 non-null uint8
481092 non-null uint8 35 color red 36 color unspecified 37 color white 38 color yellow 39 state Baden-Wuerttemberg 40 state Bavaria 41 state Berlin 42 state Brandenburg 43 state Bremen 44 state Hamburg 45 state Hesse 46 state Lower Saxony state Mecklenburg-Western Pomerania 481092 non-null uint8 48 state_North Rhine-Westphalia 481092 non-null uint8
49 state_Rhineland-Palatinate 481092 non-null uint8
50 state_Saarland 481092 non-null uint8 481092 non-null uint8 51 state_Saxony
52 state_Saxony-Anhalt 481092 non-null uint8
53 state_Schleswig-Holstein 481092 non-null uint8
54 state_Thuringia 481092 non-null uint8 51 state Saxony dtypes: float64(10), int64(4), uint8(41) memory usage: 70.2 MB Membuat Model Klasifikasi In [60]: **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=123) import pickle In [61]: with open('C:\\Users\\riska\\Downloads\\data returns\\orders train x train columns.pickle', 'wb') as fp: pickle.dump(X train.columns, fp) **Model Gausisan Naive Bayes** In [62]: from sklearn.naive_bayes import GaussianNB from sklearn.metrics import classification report from sklearn.metrics import accuracy score clf NB = GaussianNB() clf NB.fit(X train, Y train) Y pred = clf NB.predict(X test) acc = accuracy_score(Y_test, Y_pred) print("Akurasi {}".format(acc)) print(classification report(Y test, Y pred)) Akurasi 0.5549319780916451 precision recall f1-score support 0.55 0.70 0.62 49723 0.40 0.46 0.56 46496 0.55 accuracy 96219 0.55 0.55 0.54 96219 macro avg 0.54 0.55 0.55 96219 weighted avg **Model Decision Tree** In [63]: from sklearn import tree clf DT = tree.DecisionTreeClassifier() clf DT.fit(X train, Y train) Y pred = clf DT.predict(X test) acc = accuracy_score(Y_test, Y_pred) print("Akurasi {}".format(acc)) print(classification report(Y test, Y pred)) Akurasi 0.5924817343767863 precision recall f1-score support 0.61 0.60 0.60 49723 0.58 0.58 0.58 46496 0.59 96219 accuracy 0.59 0.59 0.59 macro avg 96219 weighted avg 0.59 0.59 0.59 96219 Model Random Forest In [103... | from sklearn.ensemble import RandomForestClassifier clf RF3 = RandomForestClassifier(n estimators=50, random state=123) clf_RF3.fit(X_train, Y train) Y pred = clf_RF3.predict(X_test) acc = accuracy_score(Y_test, Y_pred) print("Akurasi {}".format(acc)) print(classification_report(Y_test, Y_pred)) Akurasi 0.6455897483864933 precision recall f1-score support

 0.66
 0.66
 0.66

 0.64
 0.63
 0.63

 49723 46496 0.65 96219 accuracy 0.65 0.64 macro avq 0.65 96219 0.65 weighted avg 96219 0.65 0.65 Pembuatan model prediksi dengan menggunakan algoritma Random Forest memiliki hasil yang lebih baik daripada algoritma Naive Bayes dan Decision Tree. Selanjutnya dilakukan tuning pada parameter Random Forest atau menambah nilai estimatornya untuk mendapatkan hasil yang lebih tinggi. In [65]: **from** sklearn.ensemble **import** RandomForestClassifier clf RF2 = RandomForestClassifier(n estimators=100, random state=123) clf RF2.fit(X train, Y train) Y pred = clf RF2.predict(X test) acc = accuracy score(Y test, Y pred) print("Akurasi {}".format(acc)) print(classification report(Y test, Y pred)) Akurasi 0.6529999272492959 precision recall f1-score support 0.67 0.66 0.66 0.64 0.65 0.64 0 49723 46496 1 96219 0.65 accuracy macro avg 0.65 0.65 0.65 96219 weighted avg 0.65 0.65 0.65 96219 Ketika nilai estimatornya dinaikan menjadi 100 terdapat kenaikan pada nilai akurasi dan f1-score 1 dari 0.63 menjadi 0.64. In [66]: **from** sklearn.ensemble **import** RandomForestClassifier clf RF = RandomForestClassifier(n estimators=200, random state=123) clf RF.fit(X train, Y train) Y pred = clf RF.predict(X test) acc = accuracy_score(Y_test, Y_pred) print("Akurasi {}".format(acc)) print(classification_report(Y_test, Y_pred)) Akurasi 0.6560242779492615 precision recall f1-score support

 0.67
 0.65
 0.66
 49723

 0.64
 0.66
 0.65
 46496

 0

 accuracy
 0.66
 96219

 macro avg
 0.66
 0.66
 0.66
 96219

 weighted avg
 0.66
 0.66
 0.66
 96219

 Ketika nilai estimatornya ditambah menjadi 200 terdapat kenaikan pada nilai akurasi dan f1-score 1 dari 0.64 menjadi 0.65. Hasil dari algoritma Random Forest dengan estimator 200 ini memiliki hasil yang paling tinggi, sehingga model ini yang akan digunakan untuk melakukan prediksi. In [67]: feature importances = pd.DataFrame(clf RF.feature importances ,index = X train.columns, columns=['importance']) print(feature_importances) importance orderItemID 0.108274 customerID 0.091538 itemID 0.081366 price 0.076461 customerAge 0.072127 orderDated 0.061332 deliveryDated 0.059763 manufacturerID 0.057429 deliveryDuration 0.057111 accountAgeMonth 0.046619 deliveryMonth 0.021575 0.018589 orderMonth state North Rhine-Westphalia 0.013321 size L 0.011872 size M 0.011605 state Bavaria 0.011111 state Lower Saxony 0.011081 state Baden-Wuerttemberg 0.010664 color blue 0.010467 color black 0.010121 size XL 0.010071 size S 0.009922 color grey 0.009234 color brown 0.009073 size XXL 0.008320 state Hesse 0.008096 color green 0.007931 color red 0.007473 color purple 0.006554 state Rhineland-Palatinate 0.006477 size unsized 0.006459 state Schleswig-Holstein 0.006305 color white 0.005635 state Berlin 0.005381 state Saxony 0.004533 size XS 0.004514 state Hamburg 0.003642 state Brandenburg 0.003635 color other 0.003464 color pink 0.003225 state_Mecklenburg-Western Pomerania 0.002762 salutation Mrs 0.002742 state Thuringia 0.002728 salutation Mr 0.002373 state Bremen 0.002253 state Saxony-Anhalt 0.002251 size XXXL 0.002151 size XXS 0.002146 color orange 0.001970 state Saarland 0.001850 deliveryYear 0.001329 color yellow 0.001173 orderYear 0.001113 salutation Other 0.000030 color unspecified Cross-Validation DecisionTree In [68]: from sklearn.metrics import classification_report, accuracy score, make scorer from sklearn.model selection import cross val score from sklearn.metrics import confusion matrix def classification report with accuracy score(y true, y pred): print(classification_report(y_true, y_pred)) print(confusion matrix(y true, y pred)) return accuracy_score(y_true, y_pred) clf CV = tree.DecisionTreeClassifier() scores = cross_val_score(clf_CV, X=x, y=y, cv=4, scoring=make_scorer(classification_report_with_accuracy_score) print(scores) precision recall f1-score support 0.52 1.00 0.68 62251 0.16 0.00 0.00 58022 0.52 120273 accuracy 0.34 120273 0.34 0.50 macro avg 0.35 0.52 0.35 120273 weighted avg [[62093 158] [57991 3111 precision recall f1-score support 0.51 0.59 0.55 62250 0.47 0.40 0.43 58023 0.50 120273 accuracy 0.49 0.49 0.49 0.50 0.49 120273 macro avq 0.49 120273 weighted avg [[36620 25630] [34928 23095]] recall f1-score precision support 0 0.42 0.32 0.36 62250 0.42 58023 0.54 0.47 0.42 120273 accuracy 0.42 0.43 0.42 120273 macro avg 0.42 0.42 0.42 120273 weighted avg [[19682 42568] [26780 31243]] precision recall f1-score support 1.00 62250 0.00 0.00 0.48 1.00 0.65 58023 0.48 120273 accuracy 0.74 0.50 0.33 120273 macro avq weighted avg 0.75 0.48 0.31 120273 3 62247] 0 58023]] [0.51652491 0.49649547 0.42341174 0.48245242] Model XGBoost import xgboost as xgb clf_XGB = xgb.XGBClassifier(objective = "multi:softprob", num_class = 2, eval_metric = "mlogloss", max_depth = 24, gamma=0.1, subsample = 0.90,learning_rate=0.1, n_estimators = 100, nthread=-1) num_class adalah jumlah kelas clf_XGB.fit(X_train, Y_train) Y_pred = clf_XGB.predict(X_test) Y_pred = np.argmax(Y_pred, axis=1) acc = accuracy_score(Y_test, Y_pred) print("Akurasi {}".format(acc)) print(classification_report(Y_test, Y_pred)) Menyimpan model hasil training algoritma RandomForest In [69]: import joblib import pandas as pd joblib.dump(clf RF, 'C:\\Users\\riska\\Downloads\\data returns\\RF order.joblib') ['C:\\Users\\riska\\Downloads\\data_returns\\RF_order.joblib'] Out[69]: Implementasi Model df class = pd.read csv("C:\\Users\\riska\\Downloads\\data returns\\orders class.txt", sep=";") df class.head(5) In [71]: price customerID salutation dateOfBirth Out[71]: orderItemID orderDate deliveryDate itemID size color manufacturerID state creationDa 2013-04-0 2013-04-03 2347 89.9 12489 1963-04-26 Hesse 2012-04 43 magenta Mrs 01 2013-04-2013-04-03 2741 1963-04-26 Hesse 1 43 99.9 12489 2012-04 grey Mrs 01 2013-04-2 2013-04-03 2514 9 79.9 12489 1963-04-26 Hesse 2012-04 ecru 19 01 2013-04-2013-05-06 1963-04-26 Hesse 3 2347 42 brown 89.9 12489 Mrs 2012-04 01 2013-04-1 119.9 2690 43 12489 1963-04-26 Hesse 2012-04-4 grey Mrs 01 **Praproses** In [72]: df_class.isna().sum() Out[72]: orderItemID orderDate deliveryDate itemID manufacturerID price customerID salutation dateOfBirth state creationDate dtype: int64 In [73]: df_class.isin(['?']).sum() orderItemID Out[73]: orderDate deliveryDate itemID size color manufacturerID price customerID salutation 5169 dateOfBirth state creationDate dtype: int64 In [74]: #melihat banyak data deliveryDate yang tahunnya 1990 df class['deliveryDate'].str.contains("1990").sum() Out[74]: #melihat banyak data dateOfBirth yang tahunnya 1655 In [75]: df class['dateOfBirth'].str.contains("1655").sum() Out[75]: In [76]: df_class = df_class.replace('?', np.nan) df_class = df_class.replace('1990-12-31', np.nan) df class = df class.replace('1655-04-19', np.nan) In [77]: df_class['orderDate'] = pd.to datetime(df class['orderDate']) df_class['deliveryDate'] = pd.to_datetime(df class['deliveryDate']) df class['dateOfBirth'] = pd.to datetime(df class['dateOfBirth']) df class['creationDate'] = pd.to datetime(df class['creationDate']) In [78]: df_class.insert(2, "orderYear", np.nan) df class.insert(3, "orderMonth", np.nan) df class.insert(4, "orderDated", np.nan) df class.insert(6, "deliveryYear", np.nan) df class.insert(7, "deliveryMonth", np.nan) df class.insert(8, "deliveryDated", np.nan) df class.insert(9, "deliveryDuration", np.nan) In [79]: **from** datetime **import** timedelta df class["orderYear"] = df class["orderDate"].dt.year df class["orderMonth"] = df class["orderDate"].dt.month df class["orderDated"] = df class["orderDate"].dt.day df_class.loc[(df_class["deliveryDate"].notna()), "deliveryDuration"] = (df_class["deliveryDate"]- df_class["ord median duration = df class["deliveryDuration"].median() df class.loc[(df class["deliveryDate"].isna()), "deliveryDuration"] = median duration df class.loc[(df class["deliveryDate"].isna()), "deliveryDate"] = df class["orderDate"].loc[(df class["deliveryDate"]) df class["deliveryYear"] = df class["deliveryDate"].dt.year df class["deliveryMonth"] = df class["deliveryDate"].dt.month df class["deliveryDated"] = df class["deliveryDate"].dt.day In [80]: df_class["orderYear"] = df_class["orderYear"].astype('float') df class["orderMonth"] = df class["orderMonth"].astype('float') df class["orderDated"] = df class["orderDated"].astype('float') df class["deliveryYear"] = df class["deliveryYear"].astype('float') df class["deliveryMonth"] = df class["deliveryMonth"].astype('float') df class["deliveryDated"] = df class["deliveryDated"].astype('float') df class["deliveryDuration"] = df class["deliveryDuration"].astype('float') In [81]: #mengganti data size yang hurufnya kecil menjadi huruf kapital agar seragam df_class.loc[(df_class["size"] == "s"), "size"] = "S" df_class.loc[(df_class["size"] == "m"), "size"] = "M" df_class.loc[(df_class["size"] == "l"), "size"] = "L" df_class.loc[(df_class["size"] == "xl"), "size"] = "XL" df_class.loc[(df_class["size"] == "xs"), "size"] = "XS" df_class.loc[(df_class["size"] == "xxl"), "size"] = "XXL" df class.loc[(df class["size"] == "xxxxl"), "size"] = "XXXL" In [82]: #mengganti data size yang belakangnya terdapat tanda '+' dengan huruf untuk mengurangi variasi data df class.loc[(df class["size"] == "9+"), "size"] = "M" df class.loc[(df class["size"] == "10+"), "size"] = "L" df_class.loc[(df_class["size"] == "8+"), "size"] = "M" df class.loc[(df class["size"] == "42+"), "size"] = "L" df class.loc[(df class["size"] == "4+"), "size"] = "XS" df class.loc[(df class["size"] == "6+"), "size"] = "S" df class.loc[(df class["size"] == "7+"), "size"] = "S" df class.loc[(df class["size"] == "36+"), "size"] = "XS" df_class.loc[(df_class["size"] == "39+"), "size"] = "S" $df_{class.loc[(df_{class["size"]} == "40+"), "size"] = "M"}$ df_class.loc[(df_class["size"] == "5+"), "size"] = "XS" $df_{class.loc[(df_{class["size"] == "38+"), "size"] = "S"]}$ df class.loc[(df class["size"] == "11+"), "size"] = "L" df class.loc[(df class["size"] == "37+"), "size"] = "XS" $\label{eq:df_class} $$ df_{class["size"]} == "41+"), "size"] = "M" $$$ df_class.loc[(df_class["size"] == "3+"), "size"] = "XXS" df_class.loc[(df_class["size"] == "44+"), "size"] = "XL" df_class.loc[(df_class["size"] == "43+"), "size"] = "L" df class.loc[(df class["size"] == "13+"), "size"] = "XL" df class.loc[(df class["size"] == "45+"), "size"] = "XL" df class.loc[(df class["size"] == "46+"), "size"] = "XXL" df class.loc[(df class["size"] == "2+"), "size"] = "XXS" df class.loc[(df class["size"] == "12+"), "size"] = "XL" In [83]: #mengganti huruf menjadi angka pada atribut size untuk mengubah tipe datanya df class.loc[(df class["size"] == "S"), "size"] = "38" df class.loc[(df class["size"] == "M"), "size"] = "40" df_class.loc[(df_class["size"] == "L"), "size"] = "42" df class.loc[(df class["size"] == "XL"), "size"] = "44" df class.loc[(df class["size"] == "XS"), "size"] = "36" df class.loc[(df class["size"] == "XXS"), "size"] = "34" df class.loc[(df class["size"] == "XXL"), "size"] = "46" df class.loc[(df class["size"] == "XXXL"), "size"] = "48" df_class.loc[(df_class["size"] == "unsized"), "size"] = "0" In [84]: #mengganti tipe data size menjadi integer df class["size"] = df class["size"].astype("int64") #membuat kolom baru bertipe object untuk menampung data hasil perubahan dari atribut size df class.insert(5,"realsize", "-") In [85]: #mengganti angka menjadi huruf berdasarkan ukuran baju US df class.loc[(df class["size"] \geq = 1) & (df class["size"] \leq = 3), "realsize"] = "XXS" df_class.loc[(df_class["size"] >= 4) & (df_class["size"] <= 5), "realsize"] = "XS"</pre> $df_{class.loc[(df_{class["size"]} >= 6) & (df_{class["size"]} <= 7), "realsize"] = "S"$ df class.loc[(df class["size"] >= 8) & (df class["size"] <= 9), "realsize"] = "M"</pre> df class.loc[(df class["size"] $\gt=$ 10) & (df class["size"] $\lt=$ 11), "realsize"] = "L" $\label{eq:df_class} $$ df_class["size"] >= 12) & (df_class["size"] <= 13), "realsize"] = "XL"$ df class.loc[(df class["size"] >= 14) & (df class["size"] <= 15), "realsize"] = "XXL"</pre> $df_class.loc[(df_class["size"] >= 16) & (df_class["size"] <= 17), "realsize"] = "XXXL"$ In [86]: #mengganti angka menjadi huruf berdasarkan ukuran baju Eropa $df_{class.loc[(df_{class["size"]} \ge 18) \& (df_{class["size"]} \le 35), "realsize"] = "XXS"}$ df_class.loc[(df_class["size"] >= 36) & (df_class["size"] <= 37), "realsize"] = "XS"</pre> $df_{class.loc[(df_{class["size"]} >= 38) \& (df_{class["size"]} <= 39), "realsize"] = "S"}$ $df_class.loc[(df_class["size"] >= 40) & (df_class["size"] <= 41), "realsize"] = "M"$ $df_{class.loc[(df_{class["size"]} >= 42) \& (df_{class["size"]} <= 43), "realsize"] = "L"$ df class.loc[(df class["size"] >= 44) & (df class["size"] <= 45), "realsize"] = "XL"</pre> $df_{class.loc[(df_{class["size"]} \ge 46) \& (df_{class["size"]} <= 47), "realsize"] = "XXL"}$ df_class.loc[(df_class["size"] >= 48), "realsize"] = "XXXL" #mengubah data pada size yang 0 menjadi unsized In [87]: df class.loc[(df class["size"] == 0), "realsize"] = "unsized" #menghapus kolom size df_class = df_class.drop(['size'],axis=1) #mengganti nama kolom realsize menjadi size df class.rename(columns = {'realsize':'size'}, inplace = True) In [88]: df_class["color"] = df_class["color"].replace('basalt', 'black') df class["color"] = df class["color"].replace('ebony', 'black') df_class["color"] = df_class["color"].replace('aqua', 'blue') df_class["color"] = df_class["color"].replace('azure', 'blue') df_class["color"] = df_class["color"].replace('blau', 'blue') df class["color"] = df class["color"].replace('baltic blue', 'blue') df class["color"] = df class["color"].replace('cobalt blue', 'blue') df_class["color"] = df_class["color"].replace('dark denim', 'blue') df_class["color"] = df_class["color"].replace('dark navy', 'blue') df_class["color"] = df_class["color"].replace('darkblue', 'blue') df class["color"] = df class["color"].replace('denim', 'blue') df_class["color"] = df_class["color"].replace('navy', 'blue') df_class["color"] = df_class["color"].replace('petrol', 'blue') df_class["color"] = df_class["color"].replace('almond', 'brown') df_class["color"] = df_class["color"].replace('bronze', 'brown') df_class["color"] = df_class["color"].replace('brwon', 'brown') df_class["color"] = df_class["color"].replace('caramel', 'brown') df_class["color"] = df_class["color"].replace('cognac', 'brown') df_class["color"] = df_class["color"].replace('copper coin', 'brown') df_class["color"] = df_class["color"].replace('cortina mocca', 'brown') df_class["color"] = df_class["color"].replace('khaki', 'brown') df_class["color"] = df_class["color"].replace('mahagoni', 'brown') df_class["color"] = df_class["color"].replace('mocca', 'brown') df_class["color"] = df_class["color"].replace('ocher', 'brown') df_class["color"] = df_class["color"].replace('terracotta', 'brown') df class["color"] = df class["color"].replace('aquamarine', 'green') df_class["color"] = df_class["color"].replace('avocado', 'green') df_class["color"] = df_class["color"].replace('dark oliv', 'green') df_class["color"] = df_class["color"].replace('jade', 'green') df_class["color"] = df_class["color"].replace('mint', 'green') df_class["color"] = df_class["color"].replace('oliv', 'green') df_class["color"] = df_class["color"].replace('olive', 'green') df_class["color"] = df_class["color"].replace('turquoise', 'green') df class["color"] = df class["color"].replace('anthracite', 'grey') df class["color"] = df class["color"].replace('ash', 'grey') df_class["color"] = df_class["color"].replace('dark grey', 'grey') df_class["color"] = df_class["color"].replace('graphite', 'grey') df_class["color"] = df_class["color"].replace('iron', 'grey') df_class["color"] = df_class["color"].replace('apricot', 'orange') df_class["color"] = df_class["color"].replace('coral', 'orange') df_class["color"] = df_class["color"].replace('mango', 'orange') df class["color"] = df class["color"].replace('antique pink', 'pink') df_class["color"] = df_class["color"].replace('fuchsia', 'pink') df class["color"] = df class["color"].replace('magenta', 'pink') df_class["color"] = df_class["color"].replace('amethyst', 'purple') df_class["color"] = df_class["color"].replace('aubergine', 'purple') df_class["color"] = df_class["color"].replace('berry', 'purple') df_class["color"] = df_class["color"].replace('currant purple', 'purple')
df_class["color"] = df_class["color"].replace('hibiscus', 'purple') df_class["color"] = df_class["color"].replace('bordeaux', 'red') df_class["color"] = df_class["color"].replace('crimson', 'red') df_class["color"] = df_class["color"].replace('dark garnet', 'red') df_class["color"] = df_class["color"].replace('beige', 'white') df_class["color"] = df_class["color"].replace('champagner', 'white') df_class["color"] = df_class["color"].replace('ecru', 'white') df_class["color"] = df_class["color"].replace('ivory', 'white') df class["color"] = df class["color"].replace('silver', 'white') df_class["color"] = df_class["color"].replace('vanille', 'white') df_class["color"] = df_class["color"].replace('creme', 'yellow') df_class["color"] = df_class["color"].replace('curry', 'yellow') df class["color"] = df class["color"].replace('gold', 'yellow') df class["color"] = df_class["color"].replace('ingwer', 'yellow') df_class["color"] = df_class["color"].replace('lemon', 'yellow') #Pattern || Multicolor df_class["color"] = df_class["color"].replace('leopard', 'other') df_class["color"] = df_class["color"].replace('perlmutt', 'other') df_class["color"] = df_class["color"].replace('curled', 'other') df_class["color"] = df_class["color"].replace('aviator', 'other') df_class["color"] = df_class["color"].replace('opal', 'other') df_class["color"] = df_class["color"].replace('kanel', 'other') df_class["color"] = df_class["color"].replace('striped', 'other') df_class["color"] = df_class["color"].replace('floral', 'other') df_class["color"] = df_class["color"].replace('stained', 'other') df_class["color"] = df_class["color"].replace('ancient', 'other') df_class["color"] = df_class["color"].replace('pallid', 'other') df_class["color"] = df_class["color"].replace('nature', 'other') df_class["color"] = df_class["color"].replace('habana', 'other') #NaN value df_class.loc[(df_class["color"].isna()), "color"] = "unspecified" In [89]: | df_class['salutation'] = df_class['salutation'].replace('Family', 'Other') df class['salutation'] = df class['salutation'].replace('Company', 'Other') df class['salutation'] = df class['salutation'].replace('not reported', 'Other') In [90]: df_class.insert(19, "customerAge", np.nan) df class["customerAge"] = df class["deliveryDate"].dt.year.max() - df class["dateOfBirth"].dt.year In [91]: | df_class.loc[(df_class["customerAge"].isna()), "customerAge"] = df_class["customerAge"].median() In [92]: | upper_bound = 69 lower bound = 27 #proses penanganan outlier diatas 80 tahun df_class.loc[((df_class.customerAge > 80)), "customerAge"] = upper_bound #proses penanganan outlier dibawah 13 tahun df class.loc[((df class.customerAge < 13)), "customerAge"] = lower bound</pre> In [93]: df_class.insert(21, "accountAgeMonth", np.nan) df_class["accountAgeMonth"] = ((df_class["deliveryDate"].max() - df_class["creationDate"]) / np.timedelta64(1, In [94]: df_class = df_class.drop(['orderDate', 'deliveryDate', 'dateOfBirth', 'creationDate'],axis=1) In [95]: df_class["salutation"] = df_class["salutation"].astype('category') df class["state"] = df class["state"].astype('category') df class["color"] = df class["color"].astype('category') df class["size"] = df_class["size"].astype('category') #untuk mengubah atribut menjadi one hot method df class2 = pd.get dummies(data=df class, columns=['salutation','size','color','state']) df class2.shape In [96]: (50078, 54)Out[96]: In [97]: df_class2.info() <class 'pandas.core.frame.DataFrame'> RangeIndex: 50078 entries, 0 to 50077 Data columns (total 54 columns): Non-Null Count 50078 non-null int64 orderYear 50078 non-null float64 50078 non-null float64 orderMonth orderDated 50078 non-null float64 50078 non-null float64 deliveryYear 50078 non-null float64 deliveryMonth 50078 non-null float64 deliveryDated 50078 non-null float64 deliveryDuration itemID 8 50078 non-null int64 50078 non-null int64 9 manufacturerID 50078 non-null float64 10 price 50078 non-null int64 11 customerID 12 customerAge 50078 non-null float64 50078 non-null float64 13 accountAgeMonth 14 salutation Mr 50078 non-null uint8 15 salutation Mrs 50078 non-null uint8 16 salutation Other 50078 non-null uint8 17 size L 50078 non-null uint8 18 size M 50078 non-null uint8 19 size S 50078 non-null uint8 20 size XL 50078 non-null uint8 21 size XS 50078 non-null uint8 22 size XXL 50078 non-null uint8 23 size XXS 50078 non-null uint8 24 size XXXL 50078 non-null uint8 25 size unsized 50078 non-null uint8 26 color black 50078 non-null uint8 27 color blue 50078 non-null uint8 28 color brown 50078 non-null uint8 29 color green 50078 non-null uint8 30 color grey 50078 non-null uint8 31 color orange 50078 non-null uint8 32 color other 50078 non-null uint8 33 color pink 50078 non-null uint8 34 color purple 50078 non-null uint8 35 color red 50078 non-null uint8 50078 non-null uint8 36 color white 50078 non-null uint8 50078 non-null uint8 50078 non-null uint8 37 color yellow 38 state Baden-Wuerttemberg 39 state Bavaria 50078 non-null uint8 40 state Berlin 50078 non-null uint8
50078 non-null uint8 41 state Brandenburg 42 state Bremen 43 state Hamburg 44 state Hesse 45 state Lower Saxony 46 state Mecklenburg-Western Pomerania 50078 non-null uint8 46State_MeckTenburg-western Pomerania50078 non-null uint847state_North Rhine-Westphalia50078 non-null uint848state_Rhineland-Palatinate50078 non-null uint849state_Saarland50078 non-null uint850state_Saxony50078 non-null uint851state_Saxony-Anhalt50078 non-null uint852state_Schleswig-Holstein50078 non-null uint853state_Thuringia50078 non-null uint8 dtypes: float64(10), int64(4), uint8(40) memory usage: 7.3 MB Terdapat kolom hasil One Hot Encoding yang ada pada data training tapi tidak ada di data yang akan diprediksi, maka perlu dibuat pada urutan masing-masing kolom yang hilang (color_unspecified pada urutan 36) In [98]: df_class2.insert(36, "color_unspecified", 0) In [99]: import pickle with open ('C:\\Users\\riska\\Downloads\\data returns\\orders train x train columns.pickle', 'rb') as fp: X train column = list(pickle.load(fp)) df class2 = df class2[X train column] In [100... | #from sklearn.externals import joblib clf = joblib.load('C:\\Users\\riska\\Downloads\\data_returns\\RF_order.joblib') clf.predict(df class2)

