```
            dict_rf_imp_cols,
            dict_rf_grid,
            dict_rf_grid_pca,
            dict_rf_grid_pca_cv,
            dict_rf_grid_cv,
            dict_dt_basic,
            dict_dt,
            dict_dt_best,
            dict_dt_best_pca,
            dict_dt_best_cv,
            dict_dt_grid_cv,
            ensemble_model
           ]

pd.set_option('display.max_colwidth', None)
df_results = pd.DataFrame(results)
df_results = df_results.round(3)
df_results
```

Out[116]:

	Model	Recall	Precision	F1 Score
0	[LogisticRegression Basic]	0.002	0.333	0.005
1	[LogisticRegression Basic]	0.638	0.208	0.314
2	[Best LogisticRegression with Gridsearch]	0.638	0.208	0.314
3	[Best LogisticRegression with Gridsearch and Cross Validation]	0.641	0.209	0.315
4	[Best LogisticRegression with Gridsearch and PCA]	0.638	0.208	0.314
5	[Random Forest Basic(RF)]	0.021	0.643	0.041
6	[Random Forest Basic(RF) with important columns]	0.022	0.655	0.043
7	[Best RandomForest with GridSearch and feature importance]	0.551	0.238	0.333
8	[Best RandomForest with Gridsearch and PCA]	0.479	0.285	0.358
9	[Best RandomForest with Gridsearch, PCA and Cross Validation]	0.479	0.285	0.358
10	[Best RandomForest with Gridsearch, Feature Importance and Cross Validation(StratifiedKFold)]	0.528	0.236	0.327
11	[Decision Tree Basic]	0.230	0.234	0.232
12	[Decision Tree with GridSearch]	0.226	0.218	0.222
13	[Decision Tree with GridSearch and Feature Importance]	0.648	0.200	0.306
14	[Decision Tree with GridSearch without Feature Importance]	0.518	0.210	0.298
15	[Decision Tree with GridSearch and Cross Validation without Feature Importance]	0.675	0.190	0.297
16	[Decision Tree with GridSearch without Feature Importance]	0.572	0.198	0.293
17	[Logistic Regression - Random Forest - Decision Tree]	0.617	0.215	0.319

In [ ]:

Model selection

Wähle das beste Modell aus. Entscheide dabei selbst, welche Metrik dir am wichtigsten ist. Mit `confusion_matrix()` aus `sklearn.metrics` kannst du genau sehen, wie viele

Datenpunkte jeweils richtig und falsch klassifiziert wurden. Vielleicht hilft dir das bei deiner Entscheidung.

In [117…
```
features_train= pd.read_csv('features_train_ohe.csv')
taget_train= pd.read_csv('target_train_ohe.csv')
features_test= pd.read_csv('features_test_ohe.csv')
target_test= pd.read_csv('target_test_ohe.csv')
```

In [118…
```
#Threshold value set for important columns

print(important_columns)
```

```
13                                          VehBCost
3                                            VehOdo
4                  MMRAcquisitionAuctionAveragePrice
9                        MMRCurrentAuctionCleanPrice
5                    MMRAcquisitionAuctionCleanPrice
8                      MMRCurrentAuctionAveragePrice
11                       MMRCurrentRetailCleanPrice
10                     MMRCurrentRetailAveragePrice
6                    MMRAcquisitionRetailAveragePrice
7                       MMRAcquisitonRetailCleanPrice
12                                           VNZIP1
17                                              Day
15                                      WarrantyCost
1                                        VehicleAge
0                                           VehYear
65                       WheelType_WheelType_Covers
2                                       WheelTypeID
19                          Auction_Auction_MANHEIM
18                                    reliable_BYRNO
84                             Season_Season_Spring
61                               Color_Color_SILVER
85                             Season_Season_Summer
86                             Season_Season_Winter
62                                Color_Color_WHITE
20                              Auction_Auction_OTHER
50                                Color_Color_BLUE
74                                Size_Size_MEDIUM
16                                            Year
54                                Color_Color_GREY
82      TopThreeAmericanName_TopThreeAmericanName_GM
49                                Color_Color_BLACK
60                                  Color_Color_RED
22                               Make_Make_CHEVROLET
24                                   Make_Make_DODGE
52                                Color_Color_GOLD
81    TopThreeAmericanName_TopThreeAmericanName_FORD
25                                    Make_Make_FORD
23                                 Make_Make_CHRYSLER
75                              Size_Size_MEDIUM SUV
Name: Feature, dtype: object
```

In [ ]:
```
# Best Random Forest basierend auf wichtigen Spalten

# Initialize StandardScaler
scaler = StandardScaler()

# Fit the scaler on features_train and transform both features_train and features_t
features_train_scaled = scaler.fit_transform(features_train)
features_test_scaled = scaler.transform(features_test)

features_train_rf_scaled = features_train[important_columns]
```

```
features_test_rf_scaled = features_test[important_columns]

#Pipeline
rf_end= Pipeline([
    ('pca', PCA(n_components=None)),
    ('rf' , RandomForestClassifier(class_weight='balanced',
                                   max_depth= 10,
                                   min_samples_leaf= 1,
                                   min_samples_split= 2,
                                   n_estimators= 100,
                                   random_state=42))
    ])

#fitting
rf_end.fit(features_train_rf_scaled, target_train)

#predict
y_pred_end = rf_end.predict(features_test_rf_scaled)

#evaluate
recall_end = recall_score(target_test, y_pred_end)
precision_end = precision_score(target_test, y_pred_end)
f1_end = f1_score(target_test, y_pred_end)

print("Best RandomForestClassifier with PCA based on important columns\n----------
print('recall: ', recall_end, '\nprecision: ',precision_end, '\nf1_score: ',f1_end)
```

In [ ]:
```
confusion_matrix(target_test, y_pred_end)

#[[TN, FP],
# [FN, TP]]
```

In [ ]:
```
# 4762 TN --> Das heißt, die Anzahl des tatsächlich nicht Montagsautos, korrekt als

# 954 FP -- > Das heißt, die Anzahl des Montagsutos,die fälschlicherweise als nicht

# 461 FN --> Das heißt, die Anzahl des tatsächlich nicht Montagsautos,die fälschlic

# 385 TP --> Das heißt, die Anzahl des Montagsutos,die korrekt als Montagsautos ein
```

Glückwunsch: Du hast dein Modell gebaut und validiert. Nun kannst du alle Schritte in einer Funktion zusammenfassen, um damit die Vorhersagegüte auf dem Testdatenset zu bestimmen.

Die finale Datenpipeline

Du hast die Daten bereinigt, aufbereitet und ein Modell darauf trainiert. Kombiniere nun die jeweiligen Schritte in einer Funktion oder einer Pipeline, die ein Datenset einliest und Vorhersagen dafür erzeugt. Lösche keine Datenpunkte aus den Testdaten. Das könnte die Abschätzung der Vorhersagegüte, die das Modell im Einsatz hätte, verfälschen.

In [ ]:
```
```

In [ ]:
```
def predict_Montagsautos(csv_file, model, threshold=0.5):
    # Okuma
    train = pd.read_csv(csv_file)
    test = pd. read_csv('features_test.csv')
```

```
    # data cleaning
    train, test = clean_data(train, test)

    # feature engineering
    train, test = engineer_features(train, test)

    # datensets

    train_for_prediction = train[important_columns]

    # Tahminler
    predictions = (model.predict(train_for_prediction) > threshold).astype("int32")

    return predictions
```

Glückwunsch: Du hast nun eine Funktion, die für dich Daten einliest, aufbereitet und direkt Vorhersagen trifft. Damit kannst du den Einkäufern ein gutes Stück weiterhelfen.

Model Interpretation

Um den Einkäufern des Unternehmens dein Modell schmackhaft zu machen, solltest du ihnen erklären können, welche Eigenschaften für das Modell von Bedeutung sind.

Welche Features sind laut deinem Modell für die Vorhersage am wichtigsten?

In [ ]:

Wie ändert sich deine durchschnittliche Vorhersage, wenn du den Meilenstand `'VehOdo'` bzw. das Alter `'VehicleAge'` variierst. Würdest du das so erwarten?

In [ ]:

Abschluss des Projekts

In der Datei *features_aim.csv* findest du die Features für deine abschließende Vorhersage. Speichere nur deine vorhergesagten Werte für `'IsBadBuy'` in einer CSV-Datei namens `predictions_aim.csv`. Wie viele Käufe in den Zieldaten sollten laut deinem Modell lieber nicht getätigt werden?

In [ ]:
```
aim = pd.read_csv('features_aim.csv')
aim.info()
```

In [ ]:
```
prediction = predict_Montagsautos('features_aim.csv', rf_end)
prediction
```

In [ ]:
```
pd.DataFrame(prediction).to_csv('prediction.csv')
```

In [ ]:
```
print(len(prediction))

# vorhergesagte Anzahl der Montagsautos
count_ones = np.count_nonzero(prediction == 1)
```

```
# vorhergesagte Anzahl intakter Fahrzeuge
count_zeros = np.count_nonzero(prediction == 0)

print("die Anzahl der Klasse-1:", count_ones)
print("die Anzahl der Klasse-0:", count_zeros)
```

Glückwunsch: Du hast ein weiteres eigenständiges Data-Science-Projekt abgeschlossen! Damit bist du gut gerüstet, um Projekte selbstständig und im Team in die Tat umzusetzen!

Hast du eine Frage zu dieser Übung? Schau ins Forum, ob sie bereits gestellt und beantwortet wurde.

Fehler gefunden? Kontaktiere den Support unter support@stackfuel.com .
