In [1]:

```
#Python library
import pandas as pd
import numpy as np
from numpy import mean, std
import pickle

#import visualisation
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
import plotly


#import sklearn library
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.metrics import classification_report
from sklearn.preprocessing import StandardScaler,MinMaxScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import r2_score


#import keras library
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.wrappers.scikit_learn import KerasRegressor
from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau

%matplotlib inline
pd.set_option('display.max_columns', None)
```

```
Using TensorFlow backend.
```

In [2]:

```
class Forklift:

    def read_data(slef,data_path,sep=","):
        """
         Function to read data and return dataframe.
        :param data_path: path to read the file.
        :param sep: define separator to read file.
        :return: dataframe.

        """
        #read csv
        data=pd.read_csv(data_path,sep=sep)
        print("shape of data: {} \n".format(data.shape))
        print("columns in data: {} \n".format(data.columns))
        print("show null values in data: ")
        print(data.isnull().sum())
        print("data types for the records \n")
        print(data.dtypes)
        return data


    def process_data(slef,predictor,df,label):
        """
         Function to process data and return test and train sets.
        :param predictor: type of predictor regressor or classifier.
        :param df: dataframe.
        :param label: true label for the model.
        :return: test and train sets.

        """

        if predictor=="regressor":
            #scaler
            scaler=StandardScaler()
            #choose true labels or classes
            y=df[label]
            #create features
            data=df.copy()
            X=data.drop([label],axis=1)
            X_scaled=scaler.fit_transform(X)

            #split data into train and test sets.
            X_train, X_test, y_train, y_test=\
                                    train_test_split(X_scaled, y, test_size = 0.25)

            print('train data features and labels shape:  {} {} '.format(X_train.shape,y_tr
            print('test data features and labels shape:   {} {}'.format(X_test.shape,y_test
            return X_train, X_test, y_train, y_test

        else:

            #choose true labels or classes
            y=df[label]
            #create features
            data=df.copy()
            X=data.drop([label],axis=1)
            #split data into train and test sets.
            X_train, X_test, y_train, y_test=\
                                    train_test_split(X, y, test_size = 0.25, random_state = 21,
```

```
            return X_train, X_test, y_train, y_test


    def label_distribution(self,data, features, plot_title):

        """
         Function to plot the class label distributions.
        :param data: dataframe of interest.
        :param features: feature for distribution.
        :param plot_title: title of the plot.
        :return: plot the class label distributions.

        """

        #show the class labels distributions in the dataset.
        class_labels_df = data[features].value_counts().reset_index()
        class_labels_df.columns = [
            'label',
            'percent'
        ]

        class_labels_df['percent'] /= len(data)

        fig = px.pie(
            class_labels_df,
            names='label',
            values='percent',
            title=plot_title,
            width=800,
            height=500,

        )

        plotly.offline.plot(fig,filename='./output/'+plot_title+'.html',auto_open=False)
        fig.show()


    def feature_correlation_plot(self,df,title):
        """
         Function to plot feature correlation.
        :param df: dataframe of interest.
        :param title: title of the plot.
        :return: plot feature correlation.
        """

        # heatmap for feature correlation
        plt.figure(figsize = (10,10))
        ax=sns.heatmap(df.corr(),cmap='coolwarm', linewidths=0.5, annot=True)
        bottom, top = ax.get_ylim()
        ax.set_ylim(bottom + 0.5, top - 0.5)

        plt.title(title)
        plt.tight_layout()
        plt.savefig('./output/'+title)
        #plt.show()



    # evaluate a model
    def evaluate_model(self,X_train,y_train,model,X_test,y_test,predictor,model_name):
        """
```

```
        This function will train the model and return the model efficiency on test data.
        :param X_train,y_train,model,X_test,y_test:  Train/test features and labels.
        :return: model evaluation in the form classification report or r2_score.

        """
        if predictor=="classifier":
            model.fit(X_train, y_train)
            # Predicting the Test set results
            y_pred = model.predict(X_test)

            # define evaluation procedure
            print(classification_report(y_test, y_pred))

            # save the classifier
            with open('./trained_model/'+model_name+'.pkl', 'wb') as f:
                pickle.dump(model, f)

            return classification_report(y_test, y_pred)

        #create the model
        model.fit(X_train, y_train)
        y_pred=model.predict(X_test)

        # save the classifier
        with open('./trained_model/'+model_name+'.pkl', 'wb') as f:
            pickle.dump(model, f)

        #metrics to check the regression model on test data.
        score=r2_score(y_test, y_pred)
        #print("regressor r2_score: {}".format(r2_score(y_test, y_pred)))

        return score



    def baseline_model(self,X_train,y_train,X_test,y_test):
        """"
        This function defines the Neural Network architecture and output the trained model.
        """
        model = Sequential()
        model.add(Dense(12, input_dim=X_train.shape[1], kernel_initializer='normal', activa
        model.add(Dense(8, activation='relu'))
        model.add(Dense(1, activation='linear'))
        model.summary()
        #compile model
        model.compile(loss='mse', optimizer='adam', metrics=['mse','mae'])
        #early stop the model when there is no change in loss.
        early_stop = EarlyStopping(monitor = 'loss', min_delta = 0.001,
                                   patience = 4, mode = 'min', verbose = 1,
                                   restore_best_weights = True)

        # fit model
        history = model.fit(X_train, y_train, epochs=150, batch_size=50,  verbose=1, valida

        print(history.history.keys())
        # "Loss"
        plt.plot(history.history['loss'])
        plt.plot(history.history['val_loss'])
        plt.title('model loss')
        plt.ylabel('loss')
```

```
        plt.xlabel('epoch')
        plt.legend(['train', 'validation'], loc='upper left')
        plt.show()
        return model
```

In [3]:

```
#create class object
forklift= Forklift()
```

core data Visualisation, Preprocessing & Modeling.

In [7]:

```
#read csv data
core_data=forklift.read_data("./datasets/1_flottenuebersicht_land.csv")
core_data.head()
```

```
empfangen                          object
letzter_datenempfang               object
kostenstelle                      float64
einsatzort                        float64
fuehrerscheinklasse                 int64
freies_merkmal                    float64
dtype: object
```

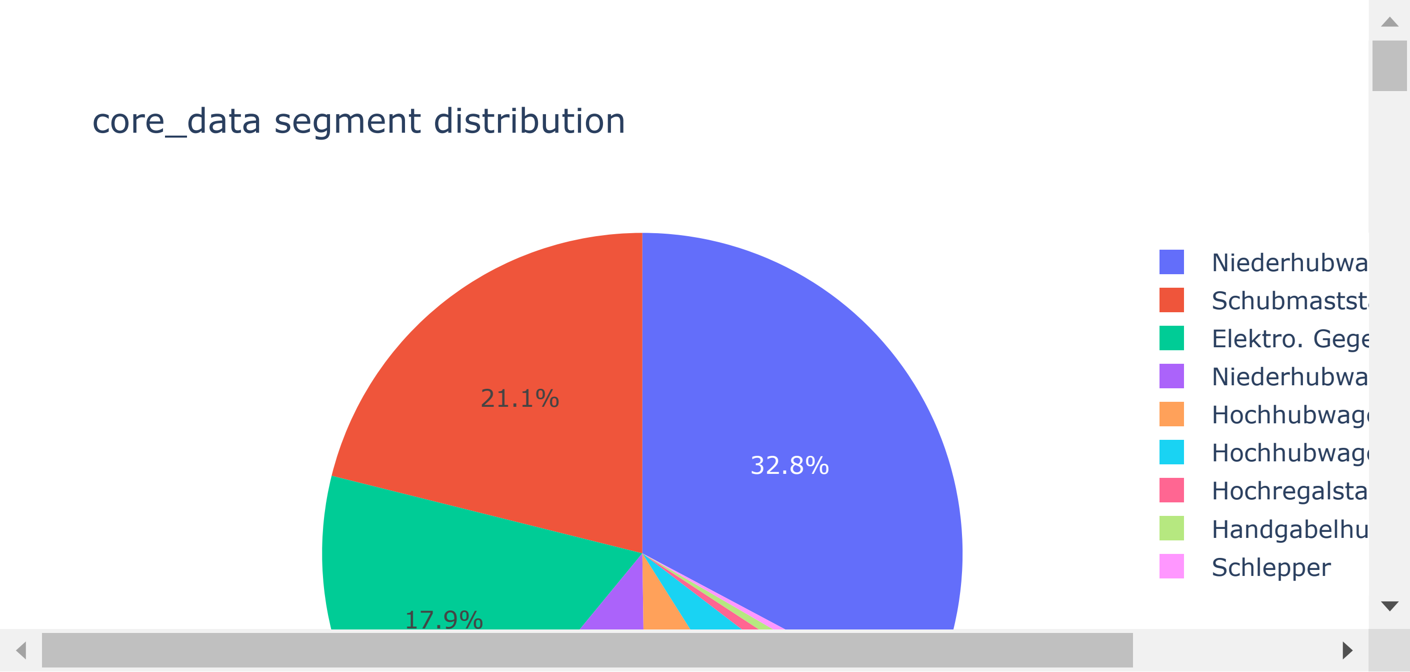
Out[7]:

	warenempfaenger	land	warenempfaenger_nummer	strasse	plz	ort	interne_nummer	equipment_numm
0	ABC	NaN	1	Weg 1	123	NaN	1.0	
1	ABC	NaN	1	Weg 1	123	NaN	2.0	
2	ABC	NaN	1	Weg 1	123	NaN	3.0	

features distribution

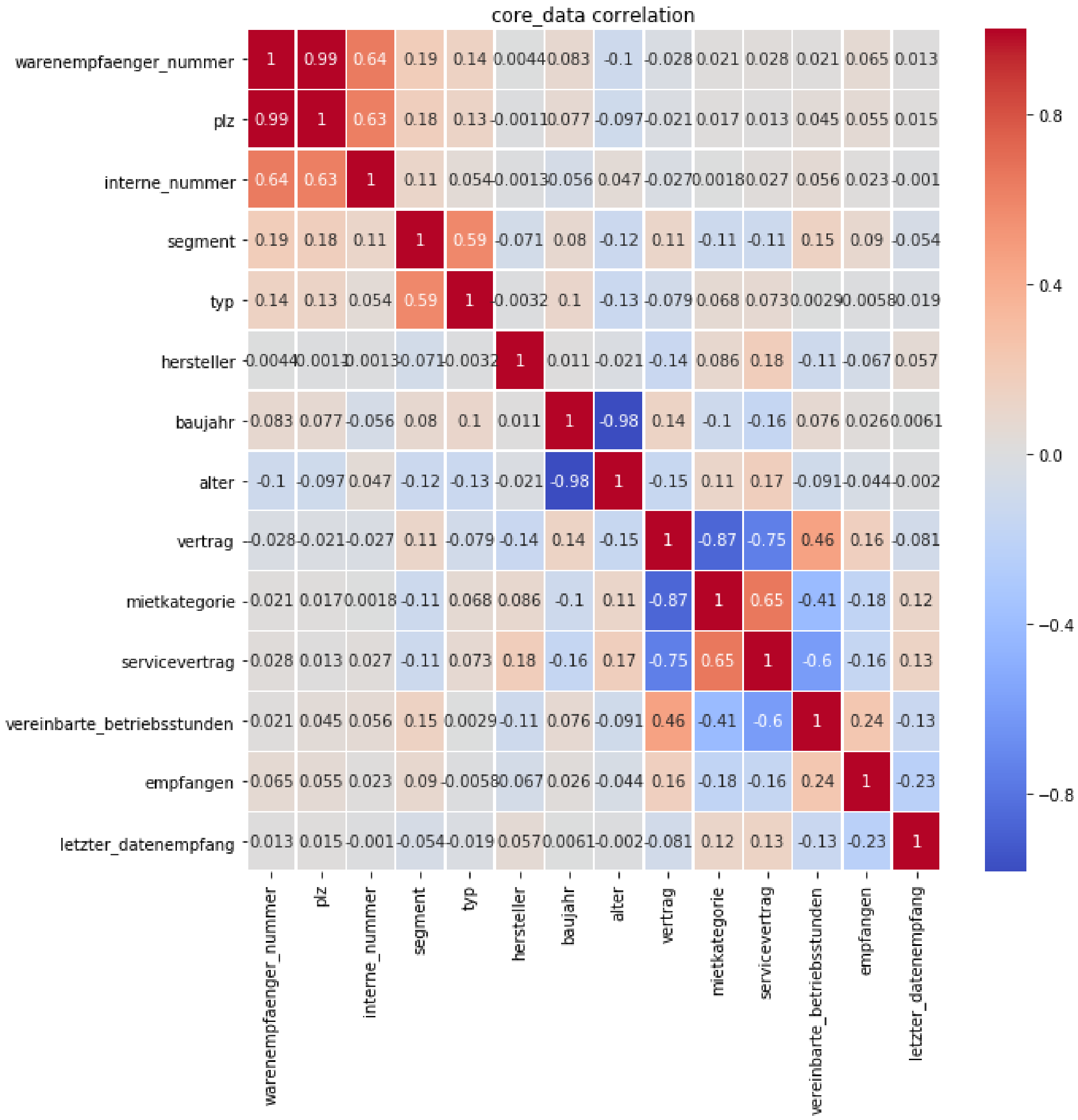
In [8]:

```
#plot future distribution
forklift.label_distribution(core_data, 'segment',' core_data segment distribution')
forklift.label_distribution(core_data, 'mietkategorie','core_data mietkategorie distributic
forklift.label_distribution(core_data, 'hersteller','core_data hersteller distribution')
forklift.label_distribution(core_data, 'vertrag','core_data vertrag distribution')
forklift.label_distribution(core_data, 'servicevertrag','core_data servicevertrag distribut
forklift.label_distribution(core_data, 'fuehrerscheinklasse','core_data fuehrerscheinklasse
```



In [23]:

```
#plot correlation
forklift.feature_correlation_plot(core_data,'core_data correlation')
```



Feature Engineering

In [9]:

```
# data preprocessing
#drop columsn with null values or columns which does make sense for example fuehrerscheinkl
core_data.drop(['warenempfaenger','land','ort','zugangsmodul','kostenstelle','einsatzort','
                'fuehrerscheinklasse','equipment_nummer','strasse'],axis=1,inplace=True)

#change lable of empfangen column to 1 for "ja" else 0
core_data['empfangen']=core_data['empfangen'].apply(lambda x: 1 if x =='Ja' else 0)

#encode the categorical values using Sklearn Labelcencoder package.
core_data[['segment','typ','hersteller','vertrag','mietkategorie','servicevertrag','letzter
core_data[['segment','typ','hersteller','vertrag','mietkategorie','servicevertrag','letzter
core_data['alter']=core_data['alter'].apply(lambda x: x.replace(',', '.')).astype(float).as

#fill nulll values with 0
core_data.fillna(0,inplace=True)
core_data.head()
```

Out[9]:

	warenempfaenger_nummer	plz	interne_nummer	segment	typ	hersteller	baujahr	alter	ve
0	1	123	1.0	0	7	0	2012	8	
1	1	123	2.0	0	7	0	2013	7	
2	1	123	3.0	0	7	0	2014	6	
3	1	123	4.0	0	7	0	2014	6	
4	1	123	5.0	0	7	0	2014	6	

Core data of the fleet : we can use this data to predict the type of Forklifts i.e segments.

```
    The problem will be classified as multiclass classification problem.
    There are 9 types of segments in total in the given dataset.
    We can use Random Forest algorithm to build our predictor.
    To validate our model, we have split data into train and test using Sklearn tr
ain_test_split.
    Since there are 9 different classes, so we have used stratify split to have sa
me ratio of class in test & test.
    We have used class_weight as "balanced_subsample" in RF model to counter the c
lass imbalancement.
    To validate our model we have used Sklearn classification_report, which gives
 per class classification.
```

Rseults:

```
    From classification report, we can observe that model works very well for thos
e classes which has more records.
    However, model also able to pick minority classes.
    Given the small amount of data, our model able to handle both majority and min
ority classes very well.
```

Imporvements:

```
    We can cross validation to hypter tune the parameters.
    To counter class imbalancement we can use synthetic over sampler such as SMOT
E.
    We can also combine minority classes with very less records and do the classif
ication.
    We can create one vs rest model and with that we can create ensembles.
```

In [10]:

```
#compute train & test sets.
X_train, X_test, y_train, y_test = forklift.process_data("classifier",core_data,'segment')
#create the model
model = RandomForestClassifier(n_estimators=500, max_depth=3,class_weight="balanced_subsamp
#train and evaluate the model
score = forklift.evaluate_model(X_train,y_train,model,X_test,y_test,'classifier','core_data
```

```
              precision    recall  f1-score   support

           0       0.82      0.97      0.89        38
           1       0.67      1.00      0.80         2
           2       0.44      0.92      0.59        12
           3       0.89      0.44      0.59        18
           4       0.50      1.00      0.67         3
           5       0.88      0.71      0.78        69
           6       1.00      1.00      1.00        24
           7       0.00      0.00      0.00         1
           8       1.00      0.98      0.99        45

    accuracy                           0.84       212
   macro avg       0.69      0.78      0.70       212
weighted avg       0.87      0.84      0.84       212
```

Cost_Per_Vehicle data Visualisation, Preprocessing & Modeling.

In [12]:

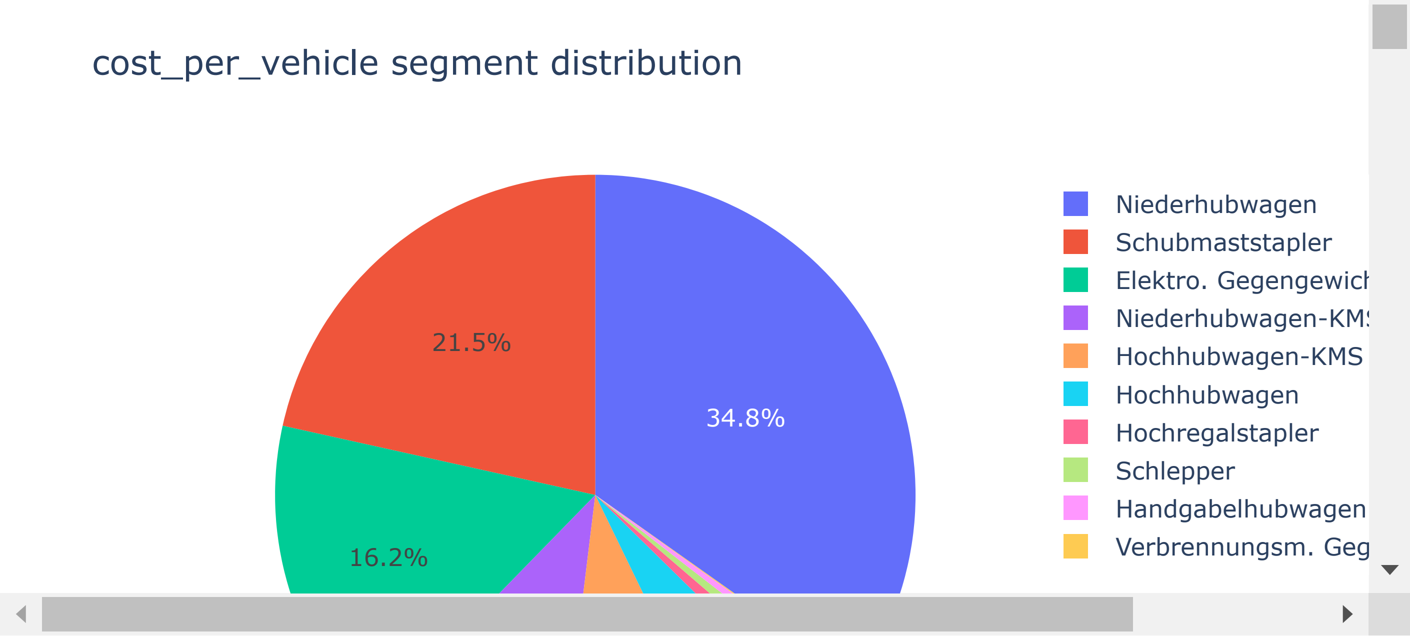
```
#read csv data
cost_per_vehicle= forklift.read_data("./datasets/2_kosten_gestapelt_land.csv",sep=";")
cost_per_vehicle.head()
```

```
shape of data: (1138, 43)

columns in data: Index(['warenempfaenger', 'warenempfaenger_nummer', 'lan
d', 'strasse',
       'postleitzahl', 'ort', 'interne_nummer', 'equipment_nummer', 'segme
nt',
       'typ', 'hersteller', 'baujahr', 'alter', 'verkaufsvertrag',
       'mietkategorie', 'vereinbarte_betriebsstunden', 'zugangsmodul',
       'kostenstelle', 'einsatzort', 'fuehrerscheinklasse', 'freies_merkma
l',
       'gesamtkosten', 'finanzierungskosten', 'mietkosten', 'servicekoste
n',
       'servicekosten_A_lohn', 'servicekosten_A_ersatzteile',
       'servicekosten_A_pauschalen', 'servicekosten_B_gewaltschaden',
       'servicekosten_B_kostenpflichtige_leistungen',
       'servicekosten_B_restriktionen', 'servicekosten_B_full_service',
       'servicekosten_B_sonstige', 'servicekosten_C_reparatur',
       'servicekosten_C_wartung', 'servicekosten_C_lservice',
```

Insights from Cost_Per_Vehicle data

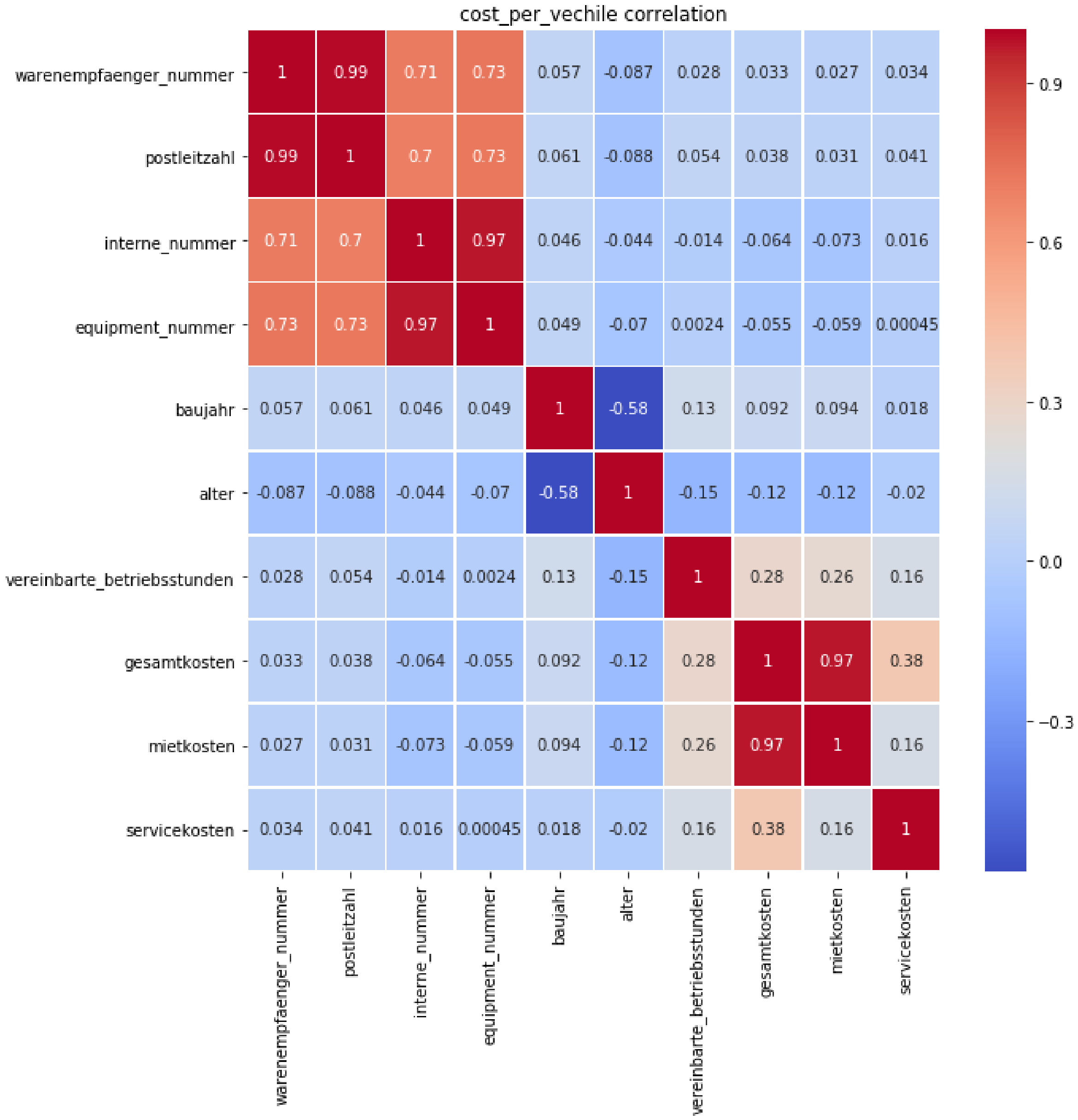
In [26]:

```
forklift.label_distribution(cost_per_vehicle, 'segment',' cost_per_vehicle segment distribu
forklift.label_distribution(cost_per_vehicle, 'mietkategorie','cost_per_vehicle mietkategor
forklift.label_distribution(cost_per_vehicle, 'hersteller',' cost_per_vehicle hersteller di
forklift.label_distribution(cost_per_vehicle, 'baujahr','cost_per_vehicle baujahr distribut
forklift.label_distribution(cost_per_vehicle, 'verkaufsvertrag','cost_per_vehicle verkaufsv
forklift.label_distribution(cost_per_vehicle, 'postleitzahl','cost_per_vehicle postleitzahl
forklift.label_distribution(cost_per_vehicle, 'strasse','cost_per_vehicle strasse distribut
```



In [28]:

```
forklift.feature_correlation_plot(cost_per_vehicle[['warenempfaenger_nummer', 'postleitzahl
       'equipment_nummer', 'segment', 'typ', 'hersteller', 'baujahr', 'alter',
       'verkaufsvertrag', 'mietkategorie', 'vereinbarte_betriebsstunden',
       'gesamtkosten', 'mietkosten', 'servicekosten']] ,'cost_per_vechile correlation')
```



	warenempfaenger_nummer	postleitzahl	interne_nummer	equipment_nummer	baujahr	alter	vereinbarte_betriebsstunden	gesamtkosten	mietkosten	servicekosten
warenempfaenger_nummer	1	0.99	0.71	0.73	0.057	-0.087	0.028	0.033	0.027	0.034
postleitzahl	0.99	1	0.7	0.73	0.061	-0.088	0.054	0.038	0.031	0.041
interne_nummer	0.71	0.7	1	0.97	0.046	-0.044	-0.014	-0.064	-0.073	0.016
equipment_nummer	0.73	0.73	0.97	1	0.049	-0.07	0.0024	-0.055	-0.059	0.00045
baujahr	0.057	0.061	0.046	0.049	1	-0.58	0.13	0.092	0.094	0.018
alter	-0.087	-0.088	-0.044	-0.07	-0.58	1	-0.15	-0.12	-0.12	-0.02
vereinbarte_betriebsstunden	0.028	0.054	-0.014	0.0024	0.13	-0.15	1	0.28	0.26	0.16
gesamtkosten	0.033	0.038	-0.064	-0.055	0.092	-0.12	0.28	1	0.97	0.38
mietkosten	0.027	0.031	-0.073	-0.059	0.094	-0.12	0.26	0.97	1	0.16
servicekosten	0.034	0.041	0.016	0.00045	0.018	-0.02	0.16	0.38	0.16	1

Feature Engineering

In [13]:

```
#drop column with null or junk values
cost_per_vehicle.drop(['warenempfaenger','land','ort','zugangsmodul','kostenstelle','einsat
                                'fuehrerscheinklasse','finanzierungskosten','strasse'],axis

#encode the categorical values using Sklearn Labelcencoder package.
cost_per_vehicle[['segment','typ','hersteller','mietkategorie','verkaufsvertrag']]= \
cost_per_vehicle[['segment','typ','hersteller','mietkategorie','verkaufsvertrag']].apply(La

#fill nulll values with 0
cost_per_vehicle.fillna(0,inplace=True)

cost_per_vehicle.head()
```

Out[13]:

	warenempfaenger_nummer	postleitzahl	interne_nummer	equipment_nummer	segment	typ
0	7	456	183.0	240	4	41
1	7	456	184.0	241	4	41
2	7	456	181.0	238	6	1
3	7	456	627.0	849	5	48
4	7	456	239.0	296	6	3

cost_per_vehicle: using this data we can predict gesamtkosten.

```
    The problem will be classified as regression problem as label to predict is co
ntinous in nature.
    Since the data is not so large to use Neural Network, we can use Random Forest
Regressor as our predictor.
    To validate our model, we have split data into train and test using Sklearn tr
ain_test_split.
     To validate our model we have use SKlearn metrics r2_Score i.e R^2 (coefficie
nt of determination)
     regression score function.
```

Rseults:

```
    Since our R^2 (coefficient of determination) value is 0.98, we can conclude th
at our predictor is fitted well.
```

Imporvements:

```
    We can cross validation to hypter tune the parameters.
    We can also compare our model with  other algorithm like Nueral Networks if we
have sufficiently large data.
```

In [14]:

```
X_train, X_test, y_train, y_test = forklift.process_data("regressor",cost_per_vehicle,"gesa

#create the model
regr = RandomForestRegressor(max_depth=7, random_state=0,n_estimators=800)
r2_score=forklift.evaluate_model(X_train,y_train,regr,X_test,y_test,'regressor','gesamtkost
print("r2_score",r2_score)
```

```
train data features and labels shape:  (853, 32) (853,)
test data features and labels shape:   (285, 32) (285,)
r2_score 0.9759210950120708
```

Operation_Interval_Of_Vechiles data Visualisation, Preprocessing & Modeling.

In [15]:

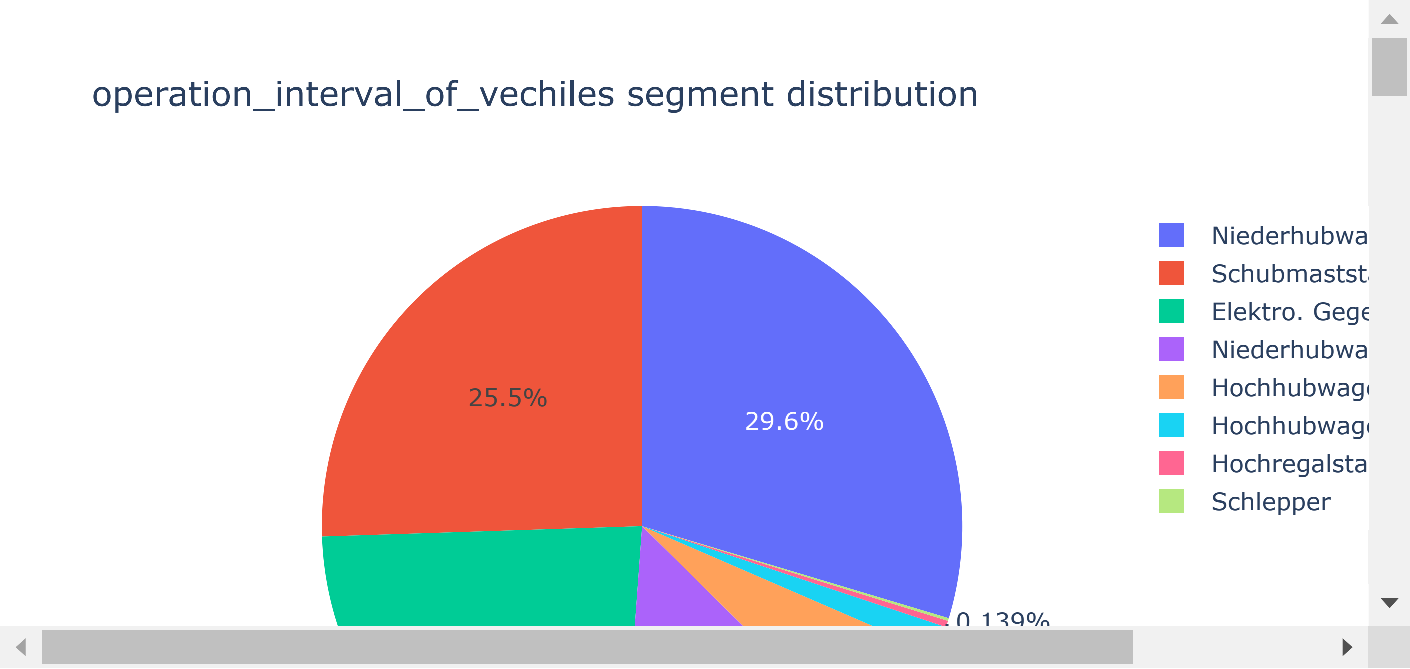
```
operation_interval_of_vechiles= forklift.read_data("./datasets/4_einsaetze_land.csv",sep=";
print(operation_interval_of_vechiles.warenempfaenger.value_counts())
operation_interval_of_vechiles.head()
```

```
shape of data: (172260, 18)

columns in data: Index(['land', 'ort', 'equipment_nummer', 'interne_numme
r', 'warenempfaenger',
       'warenempfaenger_nummer', 'segment', 'transpondertyp', 'equi_ok',
       'einsatzbeginn', 'einsatzende', 'schichttyp', 'logout', 'summe_scho
cks',
       'kostenstelle', 'einsatzort', 'fuehrerscheinklasse', 'freies_merkma
l'],
      dtype='object')

show null values in data:
land                     172260
ort                      172260
equipment_nummer              0
interne_nummer            22413
warenempfaenger               0
warenempfaenger_nummer        0
```

Insights from operation_interval_of_vechilesdata

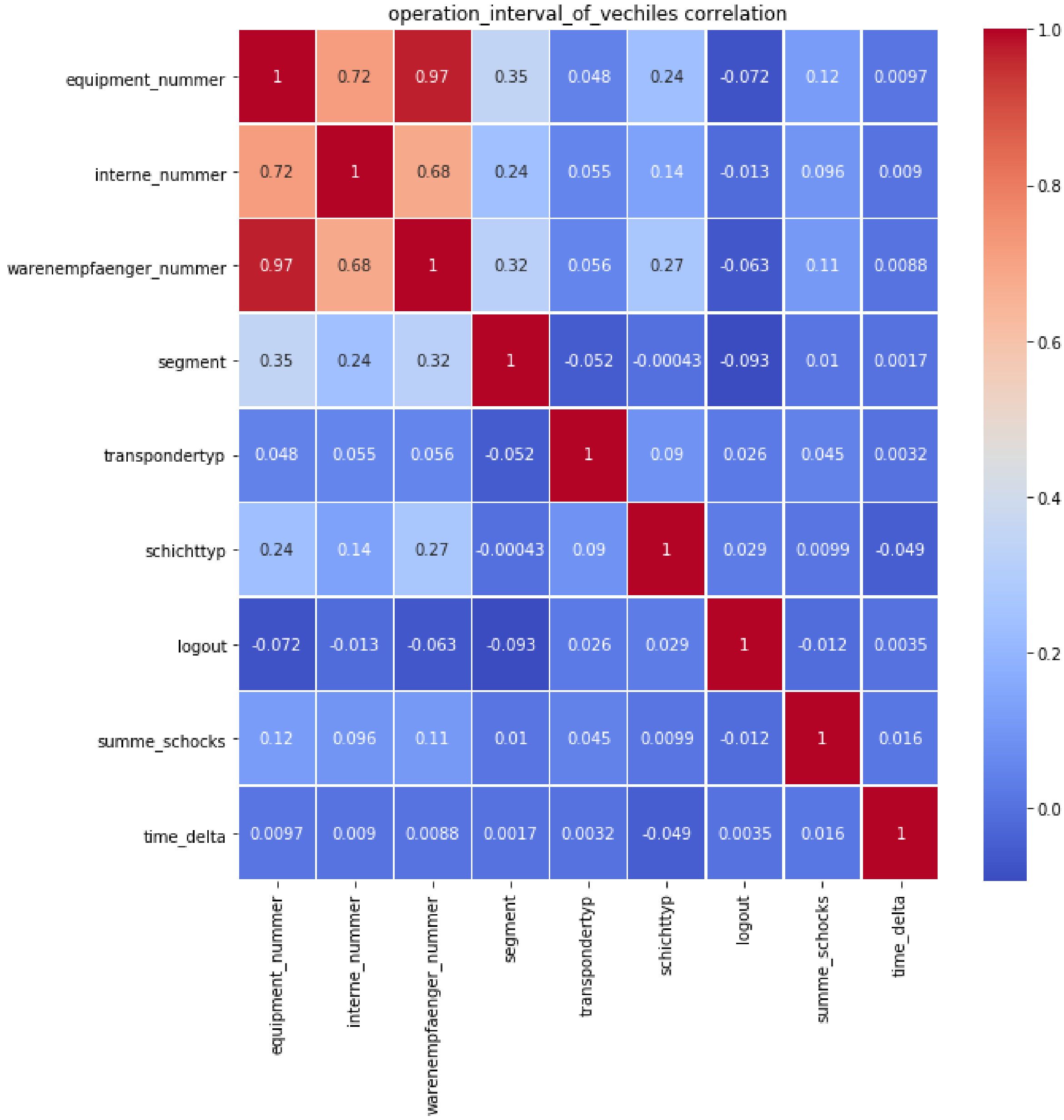
In [32]:

```
forklift.label_distribution(operation_interval_of_vechiles, 'segment',' operation_interval_
forklift.label_distribution(operation_interval_of_vechiles, 'transpondertyp','operation_int
forklift.label_distribution(operation_interval_of_vechiles, 'schichttyp',' operation_interv
forklift.label_distribution(operation_interval_of_vechiles, 'warenempfaenger_nummer','opera
forklift.label_distribution(operation_interval_of_vechiles, 'logout','operation_interval_of
```



In [35]:

```
forklift.feature_correlation_plot(operation_interval_of_vechiles,'operation_interval_of_vec
```



feature engineering

In [16]:

```
#drop columns with no values
operation_interval_of_vechiles.drop(['warenempfaenger','land','ort','kostenstelle','einsatz


#encode the categorical values using Sklearn Labelcencoder package.
operation_interval_of_vechiles[['segment','transpondertyp','schichttyp','logout']]= \
operation_interval_of_vechiles[['segment','transpondertyp','schichttyp','logout']].apply(La

#convert string to datetime
time_einsatzbeginn=pd.to_datetime(operation_interval_of_vechiles['einsatzbeginn'])
time_einsatzenden=pd.to_datetime(operation_interval_of_vechiles['einsatzende'])

#fetch duration of opeartion interval
time_delta=(time_einsatzenden-time_einsatzbeginn).astype('timedelta64[m]')

operation_interval_of_vechiles['time_delta']=time_delta
operation_interval_of_vechiles.drop(['einsatzbeginn','einsatzende'],axis=1,inplace=True)

#fill nulll values with 0
operation_interval_of_vechiles.fillna(0,inplace=True)
operation_interval_of_vechiles.head()
```

Out[16]:

	equipment_nummer	interne_nummer	warenempfaenger_nummer	segment	transpondertyp	s
0	10	10.0	1	0	0	
1	653	473.0	19	5	0	
2	373	306.0	10	4	0	
3	5	5.0	1	0	0	
4	24	22.0	1	0	0	

operation_interval_of_vechiles : There are two possible predictive models from this data

1. We can estimate the total shock i.e summe_schocks: as a prediction for shocks we can caliber the forklifs.
2. We can estimate the type of logout, using this feature we can categorize forklifts logout behavior and work on that.

The problem will be classified as regression problem as label to predict i.e is summe_schocks is continous in nature.
We have used here Neural Network using Keras Api with an input layer, a hidden layer and an output layer.
To validate our model, we have split data into train and validation set and have use MSE metrics.
We have also used Early Stopping on validation set.

Rseults:

We can use model loss plot to check the efficiency of the model.

Imporvements:

We can try with more dense layers.
Hypreparameter tuning can be done.

In [36]:

```
X_train, X_test, y_train, y_test=forklift.process_data("regressor",operation_interval_of_ve
```

```
train data features and labels shape:  (129195, 8) (129195,)
test data features and labels shape:   (43065, 8) (43065,)
```

In [35]:

```
model_nn=forklift.baseline_model(X_train,y_train,X_test,y_test)
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_10 (Dense)             (None, 12)                108       
_________________________________________________________________
dense_11 (Dense)             (None, 8)                 104       
_________________________________________________________________
dense_12 (Dense)             (None, 1)                 9         
=================================================================
Total params: 221
Trainable params: 221
Non-trainable params: 0
_________________________________________________________________
Train on 103356 samples, validate on 25839 samples
Epoch 1/150
103356/103356 [==============================] - 3s 32us/step - loss: 7.2709 - mse: 7.2709 - mae: 0.4667 - val_loss: 7.3668 - val_mse: 7.3668 - val_mae: 0.5174
```

The problem will be classified as multiclass classification problem.
There are 4 types of logout in total in the given dataset.
We can use Random Forest algorithm to build our predictor.
To validate our model, we have split data into train and test using Sklearn train_test_split.

```
    Since there are 4 different classes, so we have used stratify split to have sa
me ratio of class in test & test.
    We have used class_weight as "balanced_subsample" in RF model to counter the c
lass imbalancement.
    To validate our model we have used Sklearn classification_report, which gives
 per class classification.

    Rseults:

            From classification report, we can observe that model works  well for
 those classes which has more records.
            However, model also able to pick minority classes.

    Imporvements:

            We can cross validation to hypter tune the parameters.
            To counter class imbalancement we can use synthetic over sampler such
 as SMOTE.
            We can also combine minority classes with very less records and do the
classification.
            We can create one vs rest model and with that we can create ensembles.
```

In [17]:

```
X_train, X_test, y_train, y_test= forklift.process_data("classifier",operation_interval_of_
#create the model
model = RandomForestClassifier(n_estimators=1200, max_depth=20,class_weight="balanced_subsa
#train and evaluate the model
score = forklift.evaluate_model(X_train,y_train,model,X_test,y_test,'classifier','operation
```

```
              precision    recall  f1-score   support

           0       0.69      0.60      0.64     14705
           1       0.20      0.35      0.26      2402
           2       0.80      0.80      0.80     25646
           3       0.55      0.84      0.67       312

    accuracy                           0.71     43065
   macro avg       0.56      0.65      0.59     43065
weighted avg       0.72      0.71      0.71     43065
```

number_intencity_shocks

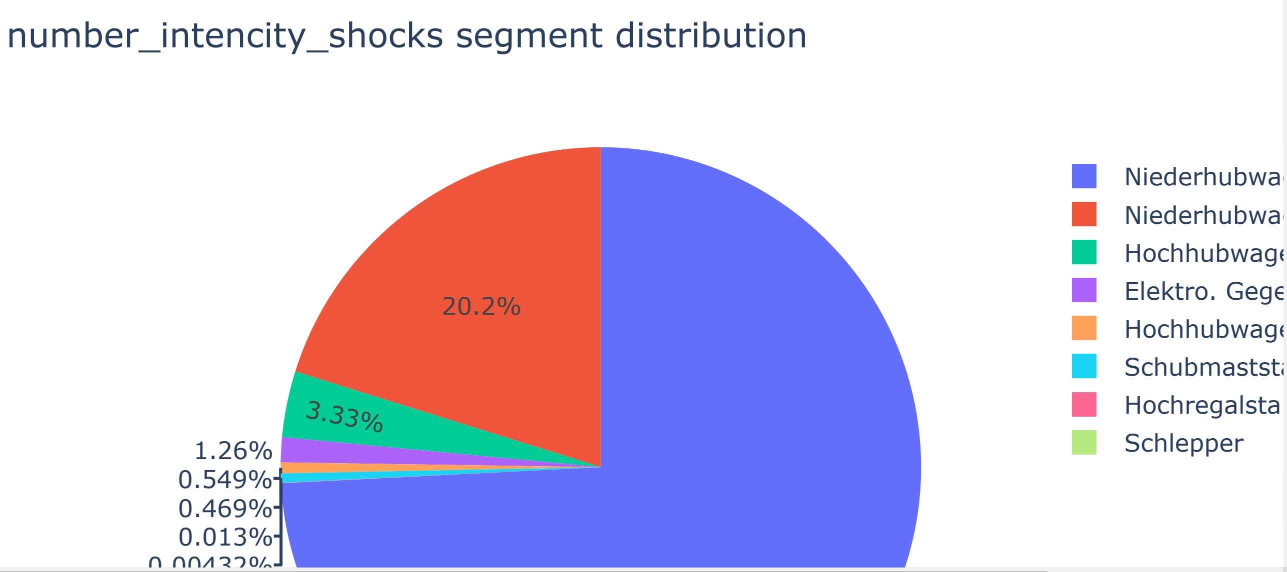
In [18]:

```
#schocklevel classification
number_intencity_shocks= forklift.read_data("./datasets/5_schocks_gestapelt_land.csv",sep=(
print(number_intencity_shocks.shape)
print(number_intencity_shocks.schocklevel.value_counts())
number_intencity_shocks.drop(['warenempfaenger','land','ort','einsatzort','freies_merkmal',

number_intencity_shocks.head()
```

```
shape of data: (46271, 24)

columns in data: Index(['land', 'ort', 'equipment_nummer', 'interne_numme
r', 'segment', 'typ',
       'warenempfaenger_nummer', 'warenempfaenger', 'strasse', 'plz',
       'mitarbeitername', 'zeitpunkt', 'schichttyp', 'schocklevel',
       'intensitaet', 'fahrzeugverhalten', 'freischaltung_durch',
       'einsatzbeginn', 'einsatzende', 'equi_ok', 'kostenstelle', 'einsatz
ort',
       'fuehrerscheinklasse', 'freies_merkmal'],
      dtype='object')

show null values in data:
land                      46271
ort                       46271
equipment_nummer              0
interne_nummer             6647
segment                       0
```

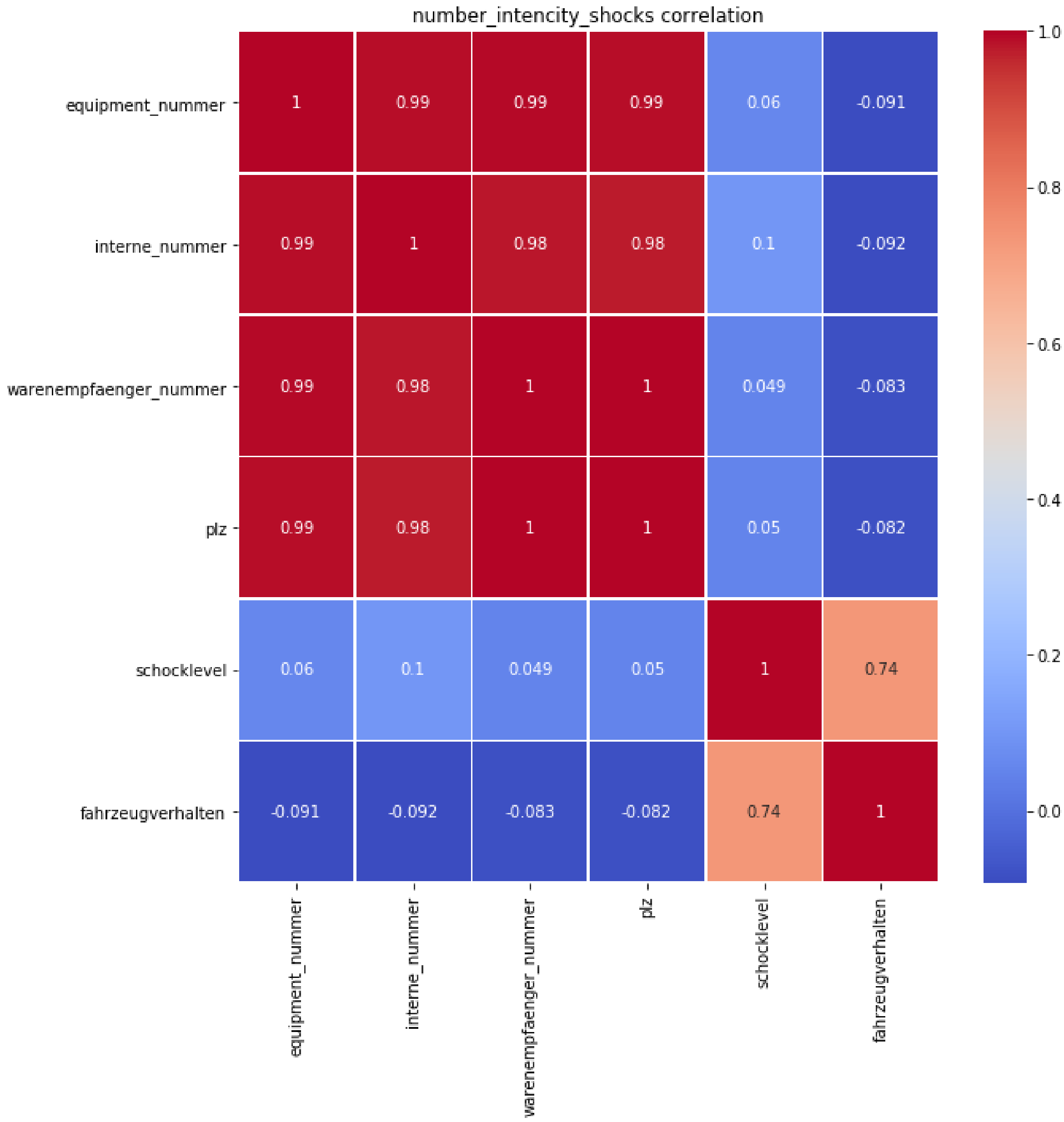
In [40]:

```
forklift.label_distribution(number_intencity_shocks, 'segment',' number_intencity_shocks se
forklift.label_distribution(number_intencity_shocks, 'typ','number_intencity_shocks typ dis
forklift.label_distribution(number_intencity_shocks, 'schocklevel',' number_intencity_shock
forklift.label_distribution(number_intencity_shocks, 'mitarbeitername','number_intencity_sh
forklift.label_distribution(number_intencity_shocks, 'fahrzeugverhalten','number_intencity_
```



In [41]:

```
forklift.feature_correlation_plot(number_intencity_shocks,'number_intencity_shocks correlat
```



Feature Engineering

In [19]:

```
#convert string to datetime
time_einsatzbeginn=pd.to_datetime(number_intencity_shocks['einsatzbeginn'])
time_einsatzenden=pd.to_datetime(number_intencity_shocks['einsatzende'])

#fetch duration of opeartion interval
time_delta=(time_einsatzenden-time_einsatzbeginn).astype('timedelta64[m]')
number_intencity_shocks['time_delta']=time_delta
number_intencity_shocks.drop(['einsatzbeginn','einsatzende','zeitpunkt'],axis=1,inplace=Tru

#encode the categorical values using Sklearn Labelcencoder package.
number_intencity_shocks[['segment','typ','schichttyp','mitarbeitername','plz']]= \
number_intencity_shocks[['segment','typ','schichttyp','mitarbeitername','plz']].apply(Label

#convert string to numeric
number_intencity_shocks['freischaltung_durch']=number_intencity_shocks['freischaltung_durch

#calculate intensity ratio
number_intencity_shocks['intensitaet']= number_intencity_shocks['intensitaet'].apply(lambda

#fill nulll values with 0
number_intencity_shocks.fillna(0,inplace=True)

number_intencity_shocks.head()
```

Out[19]:

	equipment_nummer	interne_nummer	segment	typ	warenempfaenger_nummer	plz	mitarbei
0	803	0.0	5	3	22	16	
1	803	0.0	5	3	22	16	
2	803	0.0	5	3	22	16	
3	803	0.0	5	3	22	16	
4	803	0.0	5	3	22	16	

number_intencity_shocks : we can use this data to predict the level of shock or intensity of shock, we can use this information to optimse forklifts.

```
    The problem will be classified as multiclass classification problem.
    There are 3 types of schocklevel in total in the given dataset.
    We can use Random Forest algorithm to build our predictor.
    To validate our model, we have split data into train and test using Sklearn tr
ain_test_split.
    Since there are schocklevel different classes, so we have used stratify split
 to have same ratio of
    class in test & test.
    We have used class_weight as "balanced_subsample" in RF model to counter the c
lass imbalancement.
    To validate our model we have used Sklearn classification_report, which gives
 per class classification.
```

Rseults:

```
    From classification report, we can observe that model works well in predicting
multiple classes.
```

Imporvements:

```
    We can do cross validation to hypter tune the parameters.
    To counter class imbalancement we can use synthetic over sampler such as SMOT
E.
    We can also combine minority classes with very less records and do the classif
ication.
```

In [20]:

```
X_train, X_test, y_train, y_test=forklift.process_data("classifier",number_intencity_shocks
#create the model
model = RandomForestClassifier(n_estimators=300, max_depth=10,class_weight="balanced_subsam

#train and evaluate the model
score = forklift.evaluate_model(X_train,y_train,model,X_test,y_test,'classifier','number_ir
```

```
              precision    recall  f1-score   support

           1       0.88      0.64      0.74      6226
           2       0.47      0.78      0.59      2585
           3       0.99      0.99      0.99      2757

    accuracy                           0.76     11568
   macro avg       0.78      0.81      0.77     11568
weighted avg       0.82      0.76      0.77     11568
```

opeartions_hrs_reading_dates

In [4]:

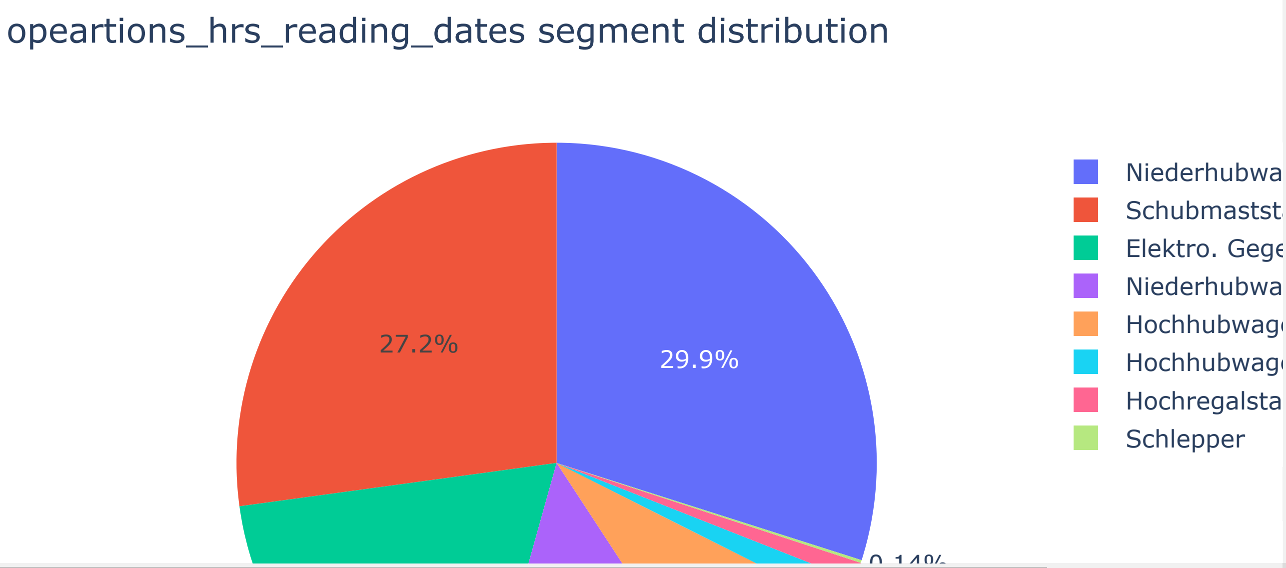
```
opeartions_hrs_reading_dates=forklift.read_data("./datasets/11_messpunkte_land.csv",sep=(";
opeartions_hrs_reading_dates.head()
```

```
shape of data: (13534, 17)

columns in data: Index(['warenempfaenger', 'warenempfaenger_nummer', 'lan
d', 'strasse', 'plz',
       'ort', 'interne_nummer', 'equipment_nummer', 'segment', 'typ',
       'hersteller', 'baujahr', 'zugangsmodul', 'zugangsmodul1', 'messdatu
m',
       'messuhrzeit', 'letzter_betriebsstundenstand'],
      dtype='object')

show null values in data:
warenempfaenger                 0
warenempfaenger_nummer          0
land                        13534
strasse                         0
plz                             0
ort                         13534
interne_nummer               1878
```

Insights from data

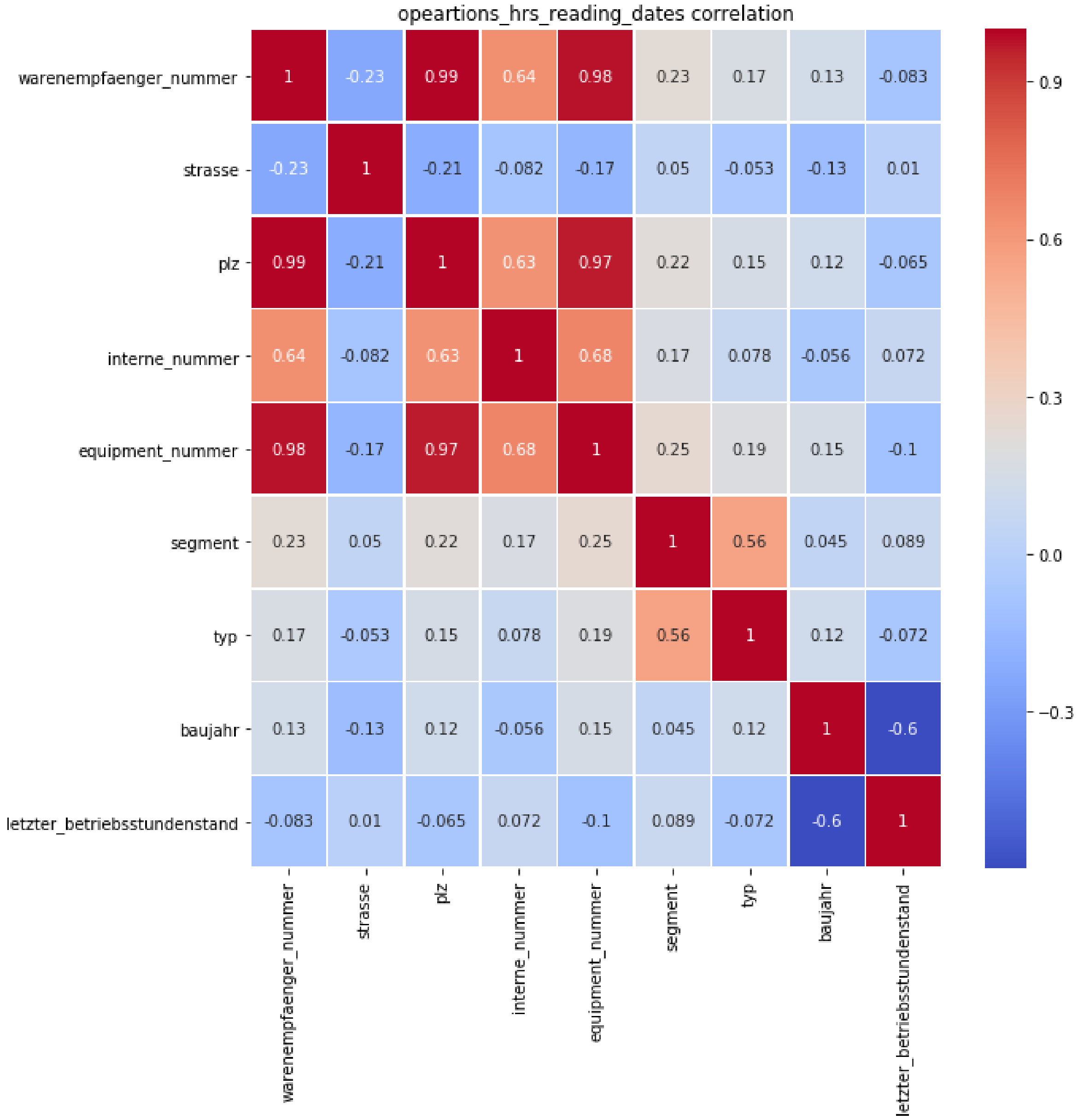
In [45]:

```
forklift.label_distribution(opeartions_hrs_reading_dates, 'segment',' opeartions_hrs_readir
forklift.label_distribution(opeartions_hrs_reading_dates, 'warenempfaenger_nummer','opearti
forklift.label_distribution(opeartions_hrs_reading_dates, 'hersteller',' opeartions_hrs_rea
```



In [7]:

```
forklift.feature_correlation_plot(opeartions_hrs_reading_dates,'opeartions_hrs_reading_date
```



Feature Engineering

In [5]:

```
opeartions_hrs_reading_dates.drop(['warenempfaenger','land','ort','zugangsmodul1','zugangsm
opeartions_hrs_reading_dates['baujahr'] = pd.DatetimeIndex(opeartions_hrs_reading_dates['ba
#encode the categorical values using Sklearn Labelcencoder package.
opeartions_hrs_reading_dates[['segment','typ','strasse','plz']]= \
opeartions_hrs_reading_dates[['segment','typ','strasse','plz']].apply(LabelEncoder().fit_tr
print(opeartions_hrs_reading_dates.shape)
#fill nulll values with 0
opeartions_hrs_reading_dates.fillna(0,inplace=True)
opeartions_hrs_reading_dates.head()
```

```
(13534, 9)
```

Out[5]:

	warenempfaenger_nummer	strasse	plz	interne_nummer	equipment_nummer	segment	typ
0	15	5	10	396.0	504	4	27
1	19	9	13	476.0	656	5	3
2	15	5	10	414.0	543	2	21
3	5	18	4	0.0	201	4	27
4	22	13	16	579.0	791	7	31

opeartions_hrs_reading_dates: using this data we can predict opearting hour level.

```
    The problem will be classified as regression problem as label to predict is co
ntinous in nature.
    we  use Random Forest Regressor as our predictor.
    To validate our model, we have split data into train and test using Sklearn tr
ain_test_split.
    To validate our model we have use SKlearn metrics r2_Score i.e R^2 (coefficien
t of determination)
     regression score function.
```

Rseults:

```
    Since our R^2 (coefficient of determination) value is 0.90, we can conclude th
at our predictor is working well.
```

Imporvements:

```
    We can do cross validation to hypter tune the parameters.
    We can also compare our model with  other algorithm like Nueral Networks if we
have sufficiently large data.
```

In [6]:

```
X_train, X_test, y_train, y_test = forklift.process_data("regressor",opeartions_hrs_reading

#create the model
regr = RandomForestRegressor(max_depth=9, random_state=0,n_estimators=800)
r2_score=forklift.evaluate_model(X_train,y_train,regr,X_test,y_test,'regressor','letzter_be
print("r2_score",r2_score)
```

```
train data features and labels shape:  (10150, 8) (10150,)
test data features and labels shape:   (3384, 8) (3384,)
r2_score 0.8932188817814902
```

In [ ]: