Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _____Информатика и системы управления_____

КАФЕДРА _____Системы обработки информации и управления_____

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ ПО ОБРАБОТКЕ И АНАЛИЗУ ДАННЫХ

НА ТЕМУ:
_____ПРЕДСКАЗАНИЕ КОЛИЧЕСТВА АВАРИЙ__
_____НА ДОРОГАХ ВЕЛИКОБРИТАНИИ_____

Студент __ИУ5-32М_____ _____ **_____Н.А. Писарчук__**
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель _____ **__Ю.Е. Гапанюк__**___
(Подпись, дата) (И.О.Фамилия)

Консультант _____ _____
(Подпись, дата) (И.О.Фамилия)

2021 г.

Введение

Правительство Великобритании собирает и публикует (обычно ежегодно) детализированную статистику о дорожно-транспортных происшествиях по всей стране. Эта статистика включает в себя географическое местоположение, погодные условия, типы транспортных средств, количество смертей и манёврах транспортных средств, что делает этот датасет очень интересным для анализа и исследования.

In [1]:
```
# This Python 3 environment comes with many helpful analytics librarie
# It is defined by the kaggle/python Docker image: https://github.com/
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" director
# For example, running this (by clicking run or pressing Shift+Enter)

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/)
# You can also write temporary files to /kaggle/temp/, but they won't

pd.set_option("display.max_columns", None)
```

```
/kaggle/input/uk-road-safety-accidents-and-vehicles/Vehicle_Informati
on.csv
/kaggle/input/uk-road-safety-accidents-and-vehicles/Accident_Informat
ion.csv
```

Описание датасета:

Датасет состоит из файлов:

- Accident_Information.csv: каждая строка в файле представляет собой уникальное дорожно-транспортное происшествие (идентифицируемое столбцом AccidentIndex), с различными свойствами, связанными с аварией, в виде столбцов. Диапазон дат: 2005-2017 гг.
- Vehicle_Information.csv: каждая строка в файле представляет участие уникального транспортного средства в уникальном дорожно-транспортном происшествии, с различными характеристиками транспортных средств и пассажиров в виде столбцов. Диапазон дат: 2004-2016 гг.
- LSOA_to_MSOA_to_Local_Authority_District_Dec_2017_Lookup.csv.

Эти файлы могут быть связаны через уникальный идентификатор дорожно-транспортного происшествия (столбец Accident_Index).

In [2]:
```
accidents = pd.read_csv('/kaggle/input/uk-road-safety-accidents-and-ve
```

```
/opt/conda/lib/python3.7/site-packages/IPython/core/interactiveshell.
py:3156: DtypeWarning: Columns (0) have mixed types.Specify dtype opt
ion on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [3]:
```
accidents.head()
```

Out[3]:

	Accident_Index	1st_Road_Class	1st_Road_Number	2nd_Road_Class	2nd_Road_Number	Ac
0	200501BS00001	A	3218.0	NaN	0.0	
1	200501BS00002	B	450.0	C	0.0	
2	200501BS00003	C	0.0	NaN	0.0	
3	200501BS00004	A	3220.0	NaN	0.0	
4	200501BS00005	Unclassified	0.0	NaN	0.0	

In [4]:
```
accidents.shape
```

Out[4]:
```
(2047256, 34)
```

Датасет содержит белее 2х миллионов строк

In [5]:
```
accidents.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2047256 entries, 0 to 2047255
Data columns (total 34 columns):
 #   Column                                       Dtype
---  ------                                       -----
 0   Accident_Index                               object
 1   1st_Road_Class                               object
 2   1st_Road_Number                              float64
 3   2nd_Road_Class                               object
 4   2nd_Road_Number                              float64
 5   Accident_Severity                            object
 6   Carriageway_Hazards                          object
 7   Date                                         object
 8   Day_of_Week                                  object
 9   Did_Police_Officer_Attend_Scene_of_Accident  float64
 10  Junction_Control                             object
```

Датасет содержит пропуски

In [6]:
```
accidents.isnull().sum()
```

Out[6]:
```
Accident_Index                                  0
1st_Road_Class                                  0
1st_Road_Number                                 2
2nd_Road_Class                             844272
2nd_Road_Number                             17593
Accident_Severity                               0
Carriageway_Hazards                             0
Date                                            0
Day_of_Week                                     0
Did_Police_Officer_Attend_Scene_of_Accident     278
Junction_Control                                0
Junction_Detail                                 0
Latitude                                      174
Light_Conditions                                0
Local_Authority_(District)                      0
Local_Authority_(Highway)                       0
Location_Easting_OSGR                         164
Location_Northing_OSGR                        164
Longitude                                     175
LSOA_of_Accident_Location                  144953
Number_of_Casualties                            0
Number_of_Vehicles                              0
Pedestrian_Crossing-Human_Control            2920
Pedestrian_Crossing-Physical_Facilities      3560
Police_Force                                    0
Road_Surface_Conditions                         0
Road_Type                                       0
Special_Conditions_at_Site                      0
Speed_limit                                    37
Time                                          156
Urban_or_Rural_Area                             0
Weather_Conditions                              0
Year                                            0
InScotland                                     53
dtype: int64
```

Преобразование типа данных колонки date

In [7]:
```
accidents['Date']= pd.to_datetime(accidents['Date'], format="%Y-%m-%d"
accidents = accidents.dropna(subset=['Time'])
```

In [8]:
```
# признак по времени - утро/рабочее время/вечер/ ночь
def feature_time(time):
    if time != 'na' :

        hour = time[0:2]

        if int(hour) >= 5 and int(hour) < 10:
            return "утро (5-10)"
        elif int(hour) >= 10 and int(hour) < 18:
            return "рабочее время (10-18)"
        elif int(hour) >= 18 and int(hour) < 23:
            return "вечер (18-23)"
        else:
            return "ночь (23-5)"

    return  0
```

In [9]:
```
accidents['Daytime'] = accidents['Time'].apply(feature_time)
accidents[['Time', 'Daytime']].head(8)
```

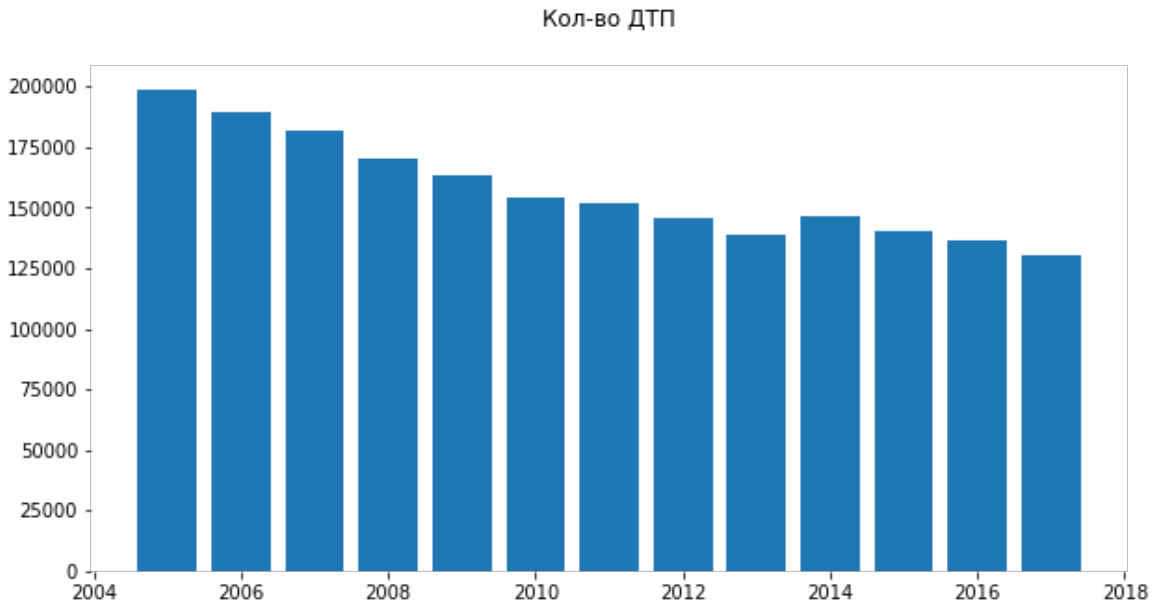
Out[9]:

	Time	Daytime
0	17:42	рабочее время (10-18)
1	17:36	рабочее время (10-18)
2	00:15	ночь (23-5)
3	10:35	рабочее время (10-18)
4	21:13	вечер (18-23)
5	12:40	рабочее время (10-18)
6	20:40	вечер (18-23)
7	17:35	рабочее время (10-18)

Увеличивается или уменьшается количество ДТП за последние годы?

In [10]:
```
# library & dataset
import seaborn as sns
import matplotlib.pyplot as plt

yearly_count = accidents['Date'].dt.year.value_counts().sort_index(asc

fig, ax = plt.subplots(figsize=(10,5))

ax.bar(yearly_count.index, yearly_count.values)
ax.set_title('\nКол-во ДТП\n')
```

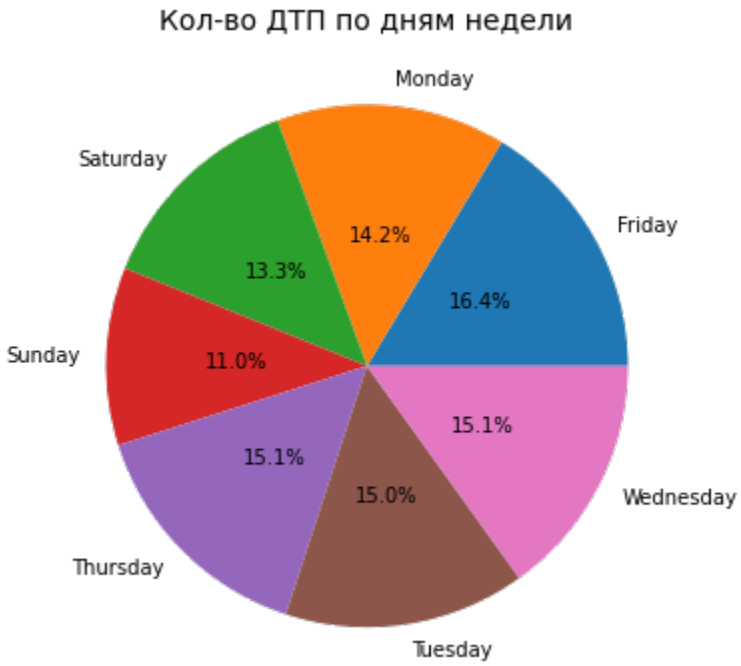
```
sns.despine(ax=ax, top=True, right=True, left=True, bottom=True);
```



Как можно видеть из графика кол-во ДТП с каждым годом. Это может быть обусловлено ужесточением санкций со стороны государства против нарушителей ПДД, с другой стороны - более ответственным отношением водителей к вождению, обновлением автомобильного парка в стране, улучшением дорожного покрытия/освещения/заграждений

Зависит ли кол-во ДТП от дня недели?

In [11]:
```
pieData = accidents.groupby("Day_of_Week")["Year"].count()

pie, ax = plt.subplots(figsize=[10,6])
labels = pieData.keys()
plt.pie(x=pieData, autopct="%.1f%%", explode=None, labels=labels, pctd
plt.title("Кол-во ДТП по дням недели", fontsize=14);
```



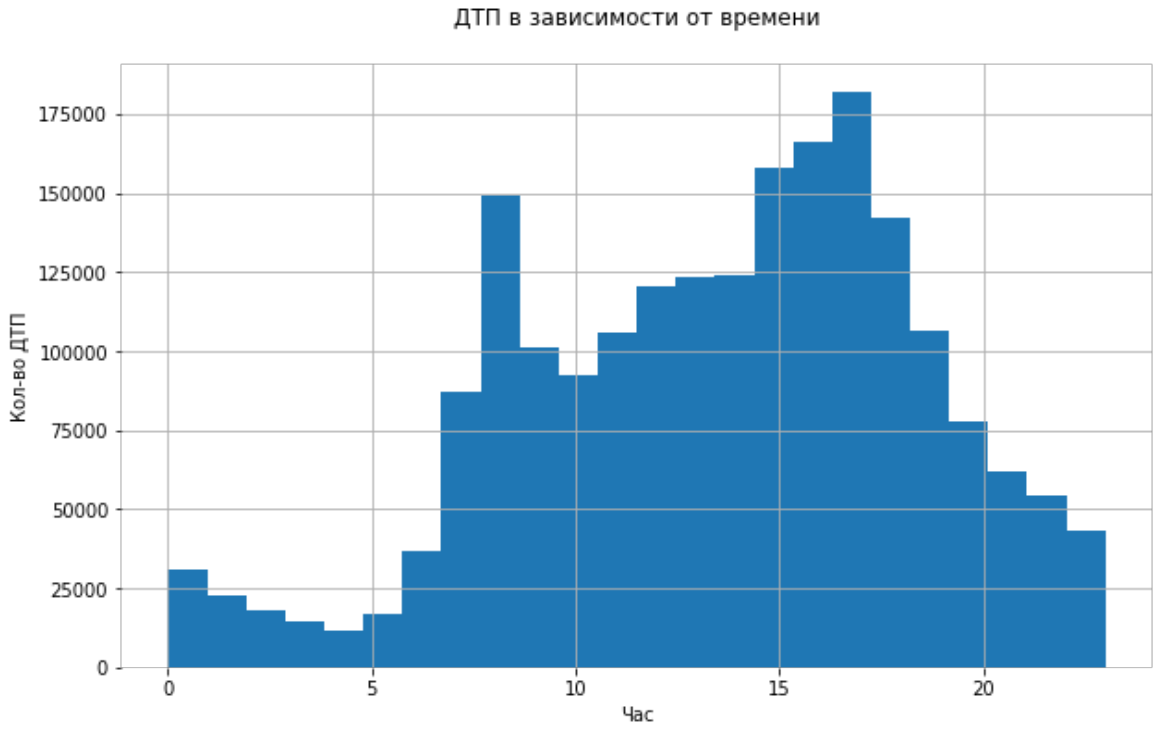
Самое большое количество ДТП случается в пятницу. Можно предположить, что это связано с увеличением траффика на дорогах - так как многие на выходные стремятся выехать за город, усталостью после рабочей недели

In [12]:
```
order = ["утро (5-10)", "рабочее время (10-18)", "вечер (18-23)", "ноч
df_sub = accidents.groupby('Daytime').size().reindex(order)

fig, ax = plt.subplots(figsize=(10, 5))

df_sub.plot(kind='barh', ax=ax)
ax.set_title('\nДТП в зависимости от времени дня\n', fontsize=14, font
ax.set(xlabel='\nОбщее кол-во ДТП', ylabel='')
```

Out[12]:
```
[Text(0.5, 0, '\nОбщее кол-во ДТП'), Text(0, 0.5, '')]
```

Самое большое количество ДТП случается в начале и в конце рабочего дня.

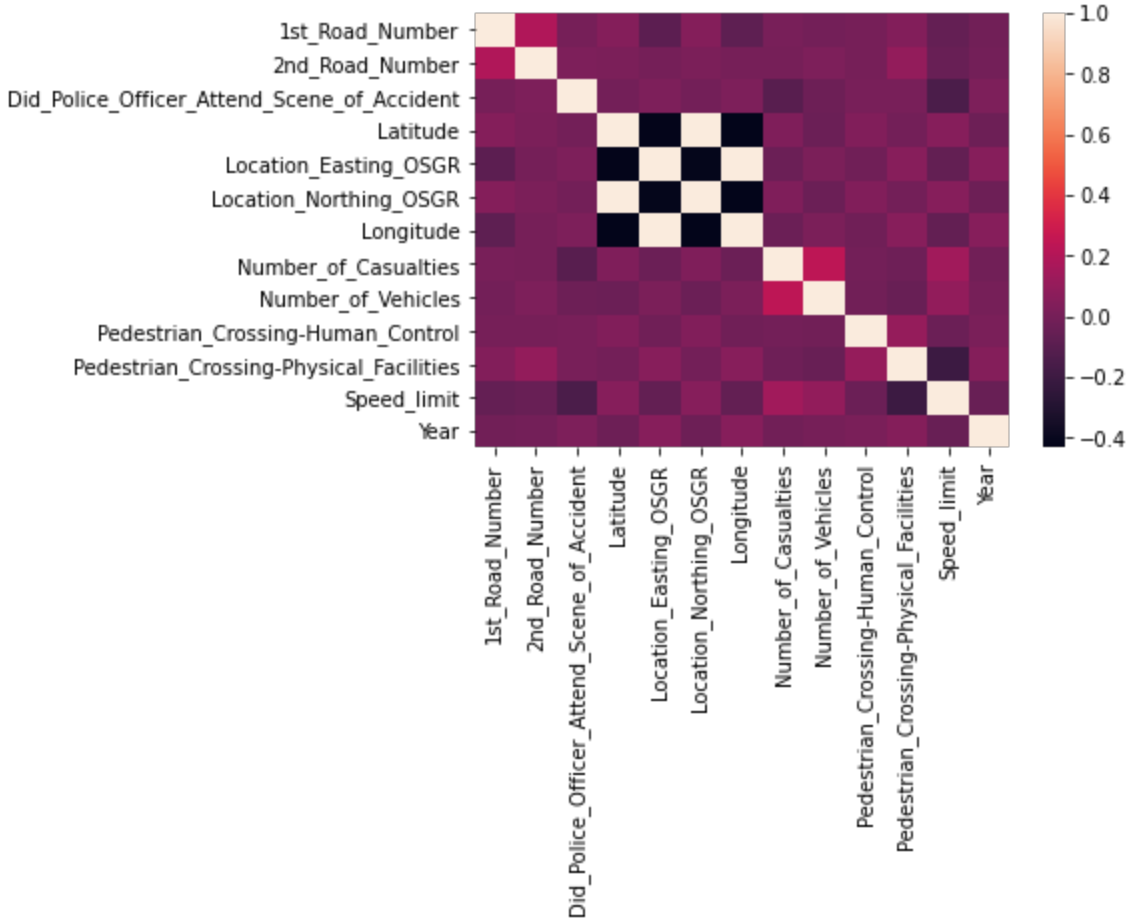
In [13]:
```
fig, ax = plt.subplots(figsize=(10,6))
accidents.Time.str[0:2].astype('int').hist(bins=24, ax=ax)
ax.set_title('\nДТП в зависимости от времени\n')
ax.set(xlabel='Час', ylabel='Кол-во ДТП')

sns.despine(top=True, right=True, left=True, bottom=True);
```



Корреляция между числовыми признаками

In [14]:
```
sns.heatmap(accidents.corr())
```

Out[14]: <AxesSubplot:>

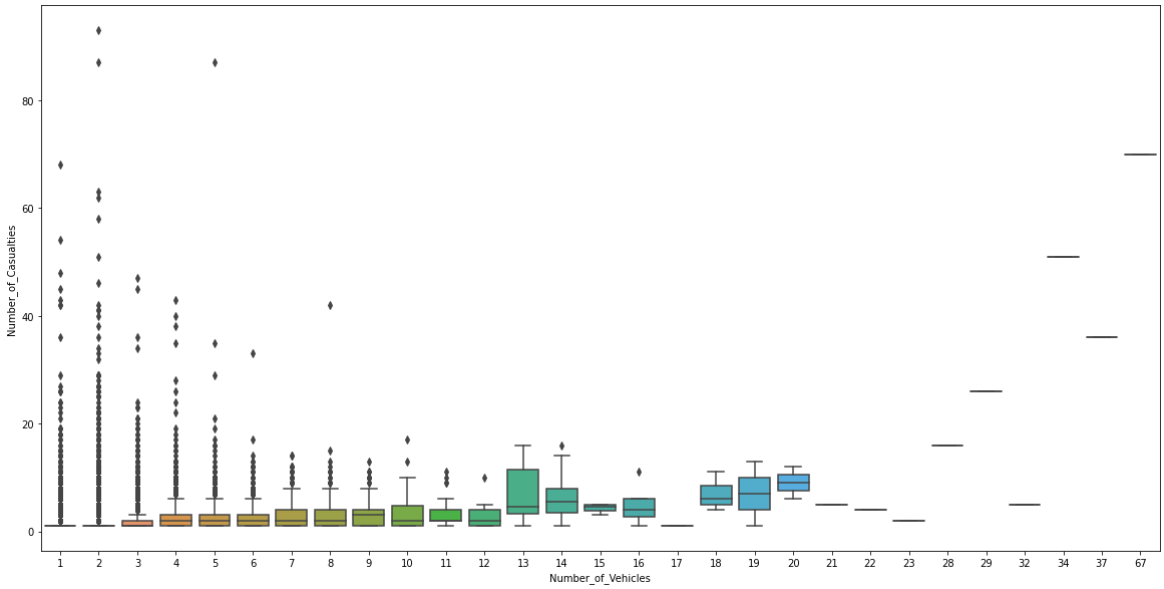


- Ограничение скорости отрицательно коррелирует с флагом приезда полиции и наличием пешеходного перехода
- Количество пострадавших связано с количеством ТС

Связь количества пострадавших с количеством ТС

In [15]:
```
plt.subplots(figsize=(20, 10))

sns.boxplot(x=accidents.Number_of_Vehicles, y=accidents.Number_of_Casu
```

Out[15]:
```
<AxesSubplot:xlabel='Number_of_Vehicles', ylabel='Number_of_Casualtie
s'>
```



- Возможно, большое количество пострадавших при количестве ТС 1-2 связано с тем, что в ДТП участвовали маршрутные ТС: вместимость маршрутного такси - 20 человек, автобуса -50
- Чем больше ТС в ДТП - тем больше жертв
- Правую часть нельзя относить к выбросам в данных, потому что, например, зимой случаются массовые ДТП при обледенении дороги

Заполнение/удаление пропусков в данных, кодирование категориальных признаков, обработка выбросов в числовых признаках

In [17]:
```
accidents = pd.read_csv('../input/uk-road-safety-accidents-and-vehicle
accidents['Date']= pd.to_datetime(accidents['Date'], format="%Y-%m-%d"

accidents['Hour'] = accidents['Time'].str[0:2]
accidents['Hour'] = pd.to_numeric(accidents['Hour'])
accidents = accidents.dropna(subset=['Hour'])
accidents['Hour'] = accidents['Hour'].astype('int')

def when_was_it(hour):
    if hour >= 5 and hour < 10:
        return "1"
    elif hour >= 10 and hour < 15:
        return "2"
    elif hour >= 15 and hour < 19:
        return "3"
```

```
    elif hour >= 19 and hour < 23:
        return "4"
    else:
        return "5"

accidents['Daytime'] = accidents['Hour'].apply(when_was_it)
accidents[['Time', 'Hour', 'Daytime']].tail()
```

```
/opt/conda/lib/python3.7/site-packages/IPython/core/interactiveshell.
py:3156: DtypeWarning: Columns (0) have mixed types.Specify dtype opt
ion on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Out[17]:

	Time	Hour	Daytime
2047251	11:30	11	2
2047252	13:00	13	2
2047253	13:30	13	2
2047254	18:00	18	3
2047255	13:00	13	2

In [18]:
```
accidents = accidents.drop(columns=['Time', 'Hour'])
```

In [19]:
```
print('Количество пропущенных значений:',
      round(accidents.isna().sum().sum()/len(accidents),3), '%')
```

```
Количество пропущенных значений: 0.495 %
```

In [20]:
```
# удаление ненужных колонок
accidents = accidents.drop(columns=['Location_Easting_OSGR', 'Location
                                    'Longitude', 'Latitude'])

# удаление пропусков
accidents = accidents.dropna()
accidents.isna().sum().sum()
```

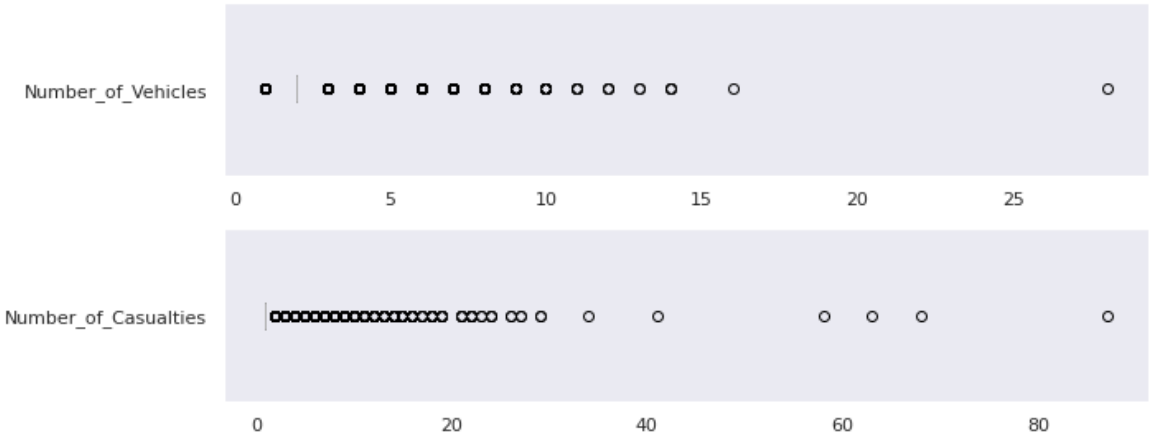
Out[20]:
```
0
```

In [31]:
```
for col in ['Accident_Severity', 'Day_of_Week', 'Daytime', 'Road_Type'
            'Urban_or_Rural_Area', 'LSOA_of_Accident_Location']:
    df[col] = df[col].astype('category')
```

Числовые признаки

In [24]:
```
num_cols = ['Number_of_Vehicles', 'Number_of_Casualties']
df = accidents[['Accident_Index', 'Accident_Severity', 'Number_of_Vehi
                'Daytime', 'Road_Type', 'Speed_limit', 'Urban_or_Rural
sns.set(style='darkgrid')
fig, axes = plt.subplots(2,1, figsize=(10,4))

for ax, col in zip(axes, num_cols):
    df.boxplot(column=col, grid=False, vert=False, ax=ax)
```

```
    plt.tight_layout();
```



In [25]:
```
# условия
condition = (df['Number_of_Vehicles'] < 6) & (df['Number_of_Casualties
df = df[condition]
print(df['Number_of_Vehicles'].value_counts())
```

```
2    784734
1    250753
3     77860
4     12980
5      2374
Name: Number_of_Vehicles, dtype: int64
```

В датасете присутствует файл LSOA_to_MSOA_to_Local_Authority_District_Dec_2017_Lookup.csv. Что это такое? LSOA - Lower Layer Super Output Area - это ГЕОГРАФИЧЕСКАЯ ОБЛАСТЬ. Области сверхвысокой производительности нижнего уровня - это географическая иерархия, разработанная для улучшения отчетности по статистике малых территорий в Англии и Уэльсе.

Области вывода нижнего уровня состоят из групп смежных областей вывода и были автоматически сгенерированы для обеспечения максимально согласованного размера населения и обычно содержат от четырех до шести областей вывода. Минимальная численность населения составляет 1000 человек, а среднее значение - 1500.

Для каждого POSTCODE в Англии и Уэльсе существует LSOA. Он доступен для Шотландии, Северной Ирландии, Нормандских островов и острова Мэн.

Местоположение может быть хорошим предсказателем количества жертв

In [28]:
```
lsoa = pd.read_csv('../input/lsoa-to-msoa/Output_Area_to_LSOA_to_MSOA_
lsoa.head(2)
```

Out[28]:

	OA11CD	OAC11CD	OAC11NM	LSOA11CD	LSOA11NM	SOAC11CD	SOAC11NM	MSO
0	E00060343	7d1	Ageing Communities and Families	E01011966	Hartlepool 006B	5b	Aspiring urban households	E02

	OA11CD	OAC11CD	OAC11NM	LSOA11CD	LSOA11NM	SOAC11CD	SOAC11NM	MSO
1	E00174083	7d1	Ageing Communities	E01011974	Hartlepool	4b	Constrained	E02

In [32]:
```
df_merged = pd.merge(df, lsoa[['LSOA11CD', 'LAD17NM']], how='left',
                     left_on='LSOA_of_Accident_Location', right_on='LS
df_merged.head(2)
```

Out[32]:

	Accident_Index	Accident_Severity	Number_of_Vehicles	Number_of_Casualties	Day_of_Week
0	200501BS00002	Slight	1	1	Wednesday
1	200501BS00002	Slight	1	1	Wednesday

In [33]:
```
df_merged = df_merged.drop(columns=['LSOA_of_Accident_Location', 'LSOA
                           .rename(columns={'LAD17NM': 'County_of_Accider
                               .astype({'County_of_Accident': 'category'}
                                   .drop_duplicates()

df_merged.head(2)
```

Out[33]:

	Accident_Index	Accident_Severity	Number_of_Vehicles	Number_of_Casualties	Day_of_Week
0	200501BS00002	Slight	1	1	Wednesday
6	200501BS00007	Slight	2	1	Thursday

In [34]:
```
num_col = ['Number_of_Vehicles']
cat_cols = ['Accident_Severity', 'Day_of_Week', 'Daytime', 'Road_Type'
            'Urban_or_Rural_Area', 'County_of_Accident']

# target
target_col = ['Number_of_Casualties']

cols = cat_cols + num_cols + target_col

# copy dataframe
df_model = df_merged[cols].copy()
df_model.shape
```

Out[34]: (1128701, 10)

In [35]:
```
dummies = pd.get_dummies(df_model[cat_cols], drop_first=True)
df_model = pd.concat([df_model[num_cols], df_model[target_col], dummie
df_model.shape
```

Out[35]: (1128701, 377)

Train-Test-Split

In [36]:
```
# features
features = df_model.drop(['Number_of_Casualties'], axis=1)

# target
target = df_model[['Number_of_Casualties']]

from sklearn.model_selection import train_test_split

# split
X_train, X_test, y_train, y_test = train_test_split(features, target,
```

Random Forest Regressor

In [37]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import RandomizedSearchCV
```

```
# create RandomForestRegressor
forest = RandomForestRegressor(random_state=4, n_jobs=-1)

# train
forest.fit(X_train, y_train)

# predict
y_train_preds = forest.predict(X_train)
y_test_preds  = forest.predict(X_test)

# evaluate
RMSE = np.sqrt(mean_squared_error(y_test, y_test_preds))
print(f"RMSE: {round(RMSE, 4)}")

r2 = r2_score(y_test, y_test_preds)
print(f"r2: {round(r2, 4)}")
```

```
RMSE: 0.7564
r2: -0.0979
```

```
# look at parameters used by our current forest
print('Parameters currently in use:\n')
pprint(forest.get_params())
```

```
Parameters currently in use:

{'bootstrap': True,
 'criterion': 'mse',
 'max_depth': None,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 10,
 'n_jobs': -1,
 'oob_score': False,
 'random_state': 4,
 'verbose': 0,
 'warm_start': False}
```

```
# create range of candidate numbers of trees in random forest
n_estimators = [100, 150]

# create range of candidate max. numbers of levels in tree
max_depth = [3, 4, 5]

# create range of candidate min. numbers of samples required to split a node
min_samples_split = [10, 15, 20]

# create dictionary with hyperparameter options
hyperparameters = dict(n_estimators=n_estimators, max_depth=max_depth, min_samples_split=min_sam
ples_split)
hyperparameters
```

```
{'n_estimators': [100, 150],
 'max_depth': [3, 4, 5],
 'min_samples_split': [10, 15, 20]}
```

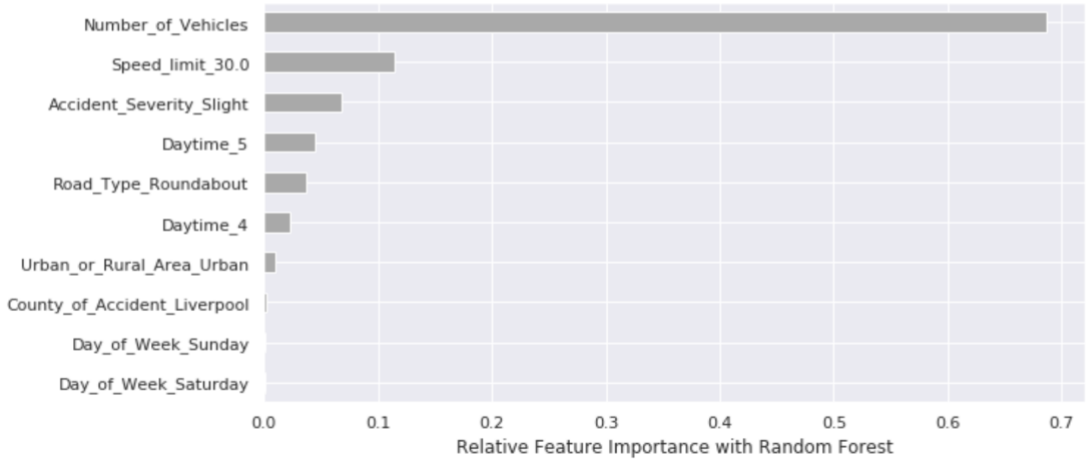
```
# create randomized search
#randomized_search = RandomizedSearchCV(forest, hyperparameters, n_jobs=-1)

# fit randomized search
#best_model = randomized_search.fit(X_train, y_train)

# view best parameters
#print(best_model.best_params_)
```

```
# view best value for specific parameter
#print(best_model.best_estimator_.get_params()['n_estimators'])
```

```
# create RandomForestRegressor with best found hyperparameters
forest = RandomForestRegressor(n_estimators=150, max_depth=5, random_state=4, n_jobs=-1)

# train
forest.fit(X_train, y_train)

# predict
y_train_preds = forest.predict(X_train)
y_test_preds  = forest.predict(X_test)

# evaluate
RMSE = np.sqrt(mean_squared_error(y_test, y_test_preds))
print(f"RMSE: {round(RMSE, 4)}")

r2 = r2_score(y_test, y_test_preds)
print(f"r2: {round(r2, 4)}")
```

```
RMSE: 0.6981
r2: 0.0649
```

```
# plot the important features
feat_importances = pd.Series(forest.feature_importances_, index=features.columns)
feat_importances.nlargest(10).sort_values().plot(kind='barh', color='darkgrey', figsize=(10,5))
plt.xlabel('Relative Feature Importance with Random Forest');
```



Вывод

В рамках выполнения данной научно-исследовательской работы была построена история о данных, которая включала в себя отображение распределений и анализ взаимосвязей исходных данных. Были проведены очистка данных и подготовка их для обучения моделей. Были выбраны, построены и обучены модели машинного обучения для предсказания потерь в ДТП в зависимости от вышеприведённых факторов.

Источники

1. Документация библиотеки seaborn https://seaborn.pydata.org/
2. Документация библиотеки pandas https://pandas.pydata.org/
3. Датасет UK Road Safety: Traffic Accidents and Vehicles https://www.kaggle.com/tsiaras/uk-road-safety-accidents-and-vehicles
4. Open Geography portalx https://geoportal.statistics.gov.uk/datasets/output-area-to-lsoa-to-msoa-to-local-authority-district-december-2017-lookup-with-area-classifications-in-great-britain/explore
5. Гапанюк Ю.Е. Методы машинного обучения – конспект лекций. https://github.com/ugapanyuk/ml_course_2021/wiki/COURSE_MMO
6. Документация библиотеки XGBoost https://xgboost.readthedocs.io/en/latest/
7. Корреляционный анализ. Конспект лекций. https://e.vyatsu.ru/pluginfile.php/462616/mod_resource/content/3/%D0%A2%D0%B5%D0%BE%D1%80%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9%20%D0%BC%D0%B0%D1%82%D0%B5%D1%80%D0%B8%D0%B0%D0%BB_%D0%BA%D0%BE%D1%80%D1%80%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9%20%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7.pdf