In [1]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-py
# 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/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all fi

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/) that gets prese
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside
```

```
/kaggle/input/playground-series-s5e10/sample_submission.csv
/kaggle/input/playground-series-s5e10/train.csv
/kaggle/input/playground-series-s5e10/test.csv
```

In [2]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Load the data:

-- Outline of the data:

> Import the data and perform the necessary preprocessing steps.

In [3]:
```
train_df=pd.read_csv('/kaggle/input/playground-series-s5e10/train.csv')
train_df.head()
```

Out[3]:

	id	road_type	num_lanes	curvature	speed_limit	lighting	weather	road_signs_present	public_roa
0	0	urban	2	0.06	35	daylight	rainy	False	Tru
1	1	urban	4	0.99	35	daylight	clear	True	Fals
2	2	rural	4	0.63	70	dim	clear	False	Tru
3	3	highway	4	0.07	35	dim	rainy	True	Tru
4	4	rural	1	0.58	60	daylight	foggy	False	Fals

In [28]:
```
def data_prerpcessing(df):
    df.info()
    print('*'*50)
    duplicate_values=df.duplicated().sum()
    print('The data contains {} duplicate values'.format(duplicate_values))
    null_values=df.isna().sum()
    print('The data contains {} null values'.format(null_values))
```

```
    print('*'*20)
    print('The data contain {} values  {} columns'.format(df.shape[0],df.shape[1]))
data_prerpcessing(train_df)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 517754 entries, 0 to 517753
Data columns (total 14 columns):
 #   Column                  Non-Null Count   Dtype  
---  ------                  --------------   -----  
 0   id                      517754 non-null  int64  
 1   road_type               517754 non-null  object 
 2   num_lanes               517754 non-null  int64  
 3   curvature               517754 non-null  float64
 4   speed_limit             517754 non-null  int64  
 5   lighting                517754 non-null  object 
 6   weather                 517754 non-null  object 
 7   road_signs_present      517754 non-null  bool   
 8   public_road             517754 non-null  bool   
 9   time_of_day             517754 non-null  object 
 10  holiday                 517754 non-null  bool   
 11  school_season           517754 non-null  bool   
 12  num_reported_accidents  517754 non-null  int64  
 13  accident_risk           517754 non-null  float64
dtypes: bool(4), float64(2), int64(4), object(4)
memory usage: 41.5+ MB
**************************************************
The data contains 0 duplicate values
The data contains id                         0
road_type                 0
num_lanes                 0
curvature                 0
speed_limit               0
lighting                  0
weather                   0
road_signs_present        0
public_road               0
time_of_day               0
holiday                   0
school_season             0
num_reported_accidents    0
accident_risk             0
dtype: int64 null values
******************
The data contain 517754 values  14 columns
```

Data Analysis And Explore Data Analysis Part:

> - Create a chart to gain insights from the data and information to understand it clearly.
> - Find the maximum, minimum and average of the speed of the data.
> - Determine the probability of high-risk lightning days.
> - Visualise the values of different road types using bar charts.
> - Visualise the maximum number of accidents occurring on different types of roads with varying numbers of lanes.

In [40]:
```
# What is the probability of the accident in public_road
public_road_accident=train_df.groupby('public_road')['accident_risk'].mean() *100
print('The percenatage of accident is not happend in the public road is {:.1f} %'.form
print('The percenatage of accident is  happend in the public road is {:.1f} %'.format(
```

```
The percenatage of accident is not happend in the public road is 34.7 %
The percenatage of accident is  happend in the public road is 35.8 %
```

In [24]:
```
# Let's understand the maximum, minimum, and average speed of the data
max_speed=train_df['speed_limit'].max()
min_speed=train_df['speed_limit'].min()
ave_speed=train_df['speed_limit'].mean()
# Let's print the values
print('Maximum speed of the vahicle is : {} km/h'.format(max_speed))
print('Minimum speed of the vahicle is : {} km/h'.format(min_speed))
print('Average speed of the vahicle is : {:.2f} km/h'.format(ave_speed))
```

```
Maximum speed of the vahicle is : 70 km/h
Minimum speed of the vahicle is : 25 km/h
Average speed of the vahicle is : 46.11 km/h
```

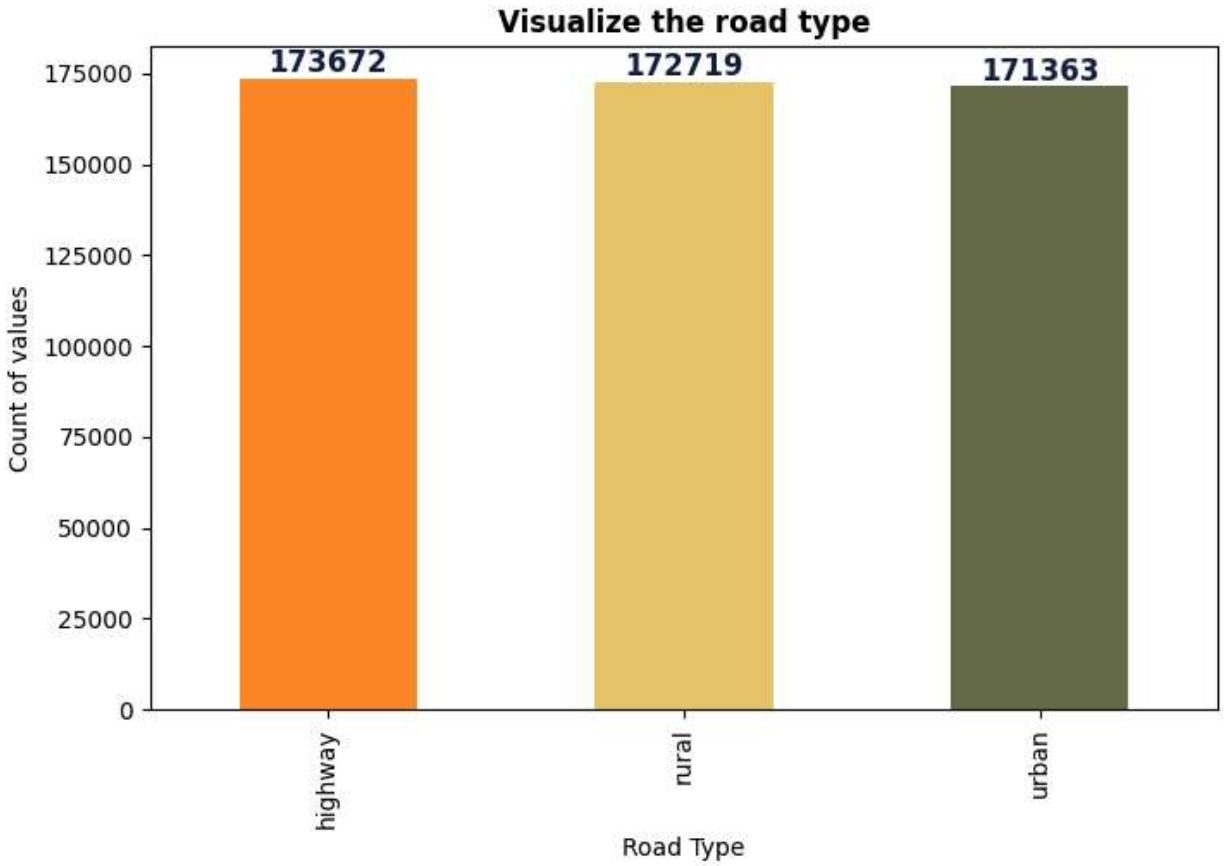
In [64]:
```
# Create the Risk percentage with day and weather
def accident_risk(value):
    if value >=0.0 and value<=0.40:
        return 'Less risk'
    elif value >=0.40 and value <=0.75:
        return 'Medium risk'
    else:
        return 'High risk'
train_df['Accident_risk']= train_df['accident_risk'].apply(accident_risk)
result_df=(train_df.groupby(['weather','time_of_day']

)['Accident_risk'].value_counts(normalize=True).mul(100).rename('Risk_percentage').res
result_df['Risk_percentage']=result_df['Risk_percentage'].round(1)
high_risk=result_df[(result_df['Accident_risk']=='High risk') & (result_df['Risk_perce
high_risk
```

Out[64]:

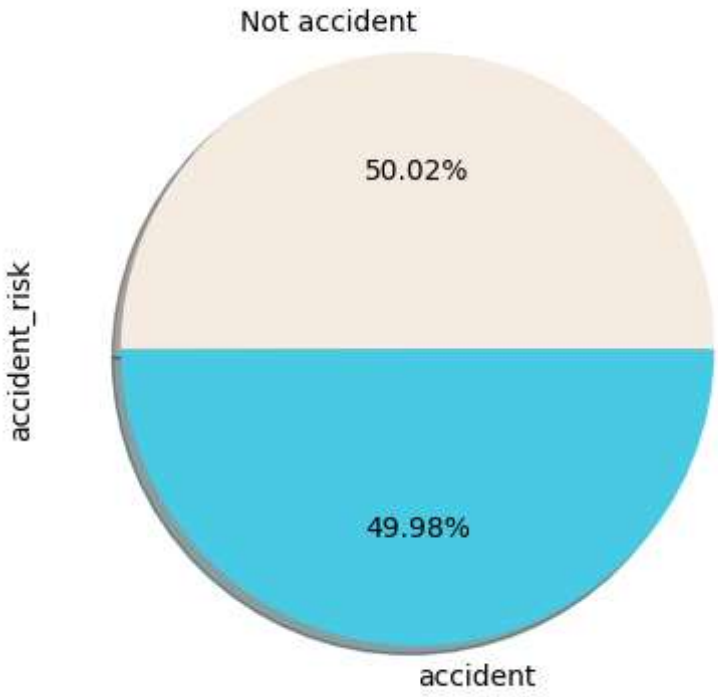
	weather	time_of_day	Accident_risk	Risk_percentage
11	foggy	afternoon	High risk	2.2
14	foggy	evening	High risk	2.1
17	foggy	morning	High risk	2.3
20	rainy	afternoon	High risk	1.6
23	rainy	evening	High risk	1.8
26	rainy	morning	High risk	1.6

In [4]:
```
# Roadtype visualization
plt.figure(figsize=(8,5))
ax=train_df['road_type'].value_counts().plot(kind='bar',color=['#fb8b24','#e9c46a','#6
ax.bar_label(ax.containers[0],fontweight='bold',fontsize=12,color='#14213d',label_type
ax.set_title('Visualize the road type',fontsize=12,fontweight='bold')
ax.set_xlabel('Road Type')
ax.set_ylabel('Count of values')
plt.show()
```



In [48]:
```
# Let's Create a pie chart
plt.figure(figsize=(10,5))
train_df.groupby('school_season')['accident_risk'].mean().plot(kind='pie',
    explode=[0,0.001],
    labels=['Not accident', 'accident'],
    colors=['#f5ebe0','#48cae4'],
    autopct='%1.2f%%',
    shadow=True)
plt.title('Accident Percentage vs School season')
plt.show()
```

Out[48]:
```
<Axes: ylabel='accident_risk'>
```



In [75]:
```
# Understanding the maximum accident cases in the different roads and different lines
max_accident_cases_in_differnt_road_lines=(
    train_df.groupby(['road_type','num_lanes'])
)['num_reported_accidents'].agg(['sum','max']).unstack()
max_accident_cases_in_differnt_road_lines.style.background_gradient(cmap='cubehelix')
```

Out[75]:

	sum				max			
num_lanes	1	2	3	4	1	2	3	4
road_type								
highway	51724	51446	50783	52578	5	6	6	6
rural	50815	51131	50351	51875	6	6	6	5
urban	50595	50869	51227	51682	6	7	7	6

In [60]:
```
total_accident_cases_in_days=pd.DataFrame(train_df.groupby(['lighting','weather'])['nu
total_accident_cases_in_days.style.background_gradient(cmap='gist_stern_r')
```

Out[60]:

weather	clear	foggy	rainy
lighting			
daylight	64720	74424	68435
dim	64577	76120	70586
night	75397	65256	55561

Machine Learning Model Building:

- This is a regression problem where we aim to find the best-fit line, as well as calculate the R^2 score and mean squared error.
- First, we import the necessary libraries for the machine learning model. Next, I convert categorical variables into numerical ones using a label encoder. Finally, I divide the data into independent variables (X) and the dependent variable (Y).
- Split the data into training and testing sets, using 20% of the data for testing. Then, write a function for building a machine learning model that returns metrics such as R^2 score and mean squared error.
- Then I applied different regression algorithms such as Linear Regression, Random Forest, Decision Tree, and XGBoost. After that, I performed hyperparameter tuning.

In [6]:
```
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score,mean_squared_error,mean_absolute_error
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import LabelEncoder,MinMaxScaler

# preprocessing step converts the categroical to numerical
def label_encoder(text_coulumn):
    label=LabelEncoder()
    return label.fit_transform(text_coulumn)
cat=train_df.select_dtypes(include=['object','bool']).copy()
for column in cat.columns:
    train_df[column]=label_encoder(train_df[column])

# Split the data into dependent and indepent values
X=train_df.drop('accident_risk',axis=1)
y=train_df['accident_risk']
# Normalise the data
def normalize_data(df):
    scaler=MinMaxScaler()
    return scaler.fit_transform(df)
X=normalize_data(X)
#Split the data into train and test
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)


# Let's create the model buliding

def model_buliding(model,X_train,X_test,y_train,y_test):
    # Let's fit the model
    model.fit(X_train,y_train)
    # Prediction model
    model_pred=model.predict(X_test)
    # mean_squared_error
    mse=mean_squared_error(y_test,model_pred)
    # Root mean squared error
    rmse=np.sqrt(mse)
    # Mean abousulte error
    mae=mean_absolute_error(y_test,model_pred)
    # r2_score
    score=r2_score(y_test,model_pred)
    print('mean squared error : {}'.format(mse))
```

```
    print('mean abosulte error : {}'.format(mae))
    print('Root mean squared error : {}'.format(rmse))
    print('R2 score : {}'.format(score))
# Apply the Linear Regression algorithms
linear=LinearRegression()
model_buliding(linear,X_train,X_test,y_train,y_test)
```

```
mean squared error : 0.007822333727292028
mean abosulte error : 0.0708090973358205
Root mean squared error : 0.08844395811638027
R2 score : 0.7167072760552073
```

In [7]:
```
random=RandomForestRegressor(n_estimators=100,criterion='squared_error',random_state=4
model_buliding(random,X_train,X_test,y_train,y_test)
```

```
mean squared error : 0.0033674749455823703
mean abosulte error : 0.04500622591766377
Root mean squared error : 0.058029948695327745
R2 score : 0.8780439209821177
```

In [11]:
```
from xgboost import XGBRegressor
xgb_model=XGBRegressor()
model_buliding(xgb_model,X_train,X_test,y_train,y_test)
```

```
mean squared error : 0.0031779898126412616
mean abosulte error : 0.04374151718859403
Root mean squared error : 0.05637366240223587
R2 score : 0.8849062924085169
```

Hyperparameter turning

In [9]:
```
params={
    'fit_intercept':[True,False],
    'copy_X': [True,False],
    'n_jobs':[10,15,20]

}
linear=LinearRegression()
linear_grid=GridSearchCV(linear,param_grid=params,n_jobs=10,scoring='r2')
model_buliding(linear_grid,X_train,X_test,y_train,y_test)
```

```
mean squared error : 0.007822333727292028
mean abosulte error : 0.0708090973358205
Root mean squared error : 0.08844395811638027
R2 score : 0.7167072760552073
```

In [ ]:
```
params={
    'n_estimators':[5,7,9],
    'criterion':["squared_error"],
    'max_depth':[2,4,6]
}

random_grid=GridSearchCV(random,param_grid=params,scoring='r2')
model_buliding(random_grid,X_train,X_test,y_train,y_test)
```

In [14]:
```
from sklearn.ensemble import VotingRegressor
voting=VotingRegressor(
    estimators=[('lr',linear_grid),('xgb',xgb_model),('random',random)]
```

```
)
model_buliding(voting,X_train,X_test,y_train,y_test)
```

```
mean squared error : 0.003690477523586603
mean abosulte error : 0.04757053983098168
Root mean squared error : 0.06074930060162506
R2 score : 0.8663460973716584
```

In [23]:
```
from sklearn.tree import DecisionTreeRegressor
tree=DecisionTreeRegressor()
model_buliding(tree,X_train,X_test,y_train,y_test)
```

```
mean squared error : 0.00678109433998706
mean abosulte error : 0.06360508348543231
Root mean squared error : 0.08234740032318603
R2 score : 0.7544166799993354
```

In [15]:
```
test=pd.read_csv('/kaggle/input/playground-series-s5e10/test.csv')
test.head()
```

Out[15]:

	id	road_type	num_lanes	curvature	speed_limit	lighting	weather	road_signs_present	publi
0	517754	highway	2	0.34	45	night	clear	True	
1	517755	urban	3	0.04	45	dim	foggy	True	
2	517756	urban	2	0.59	35	dim	clear	True	
3	517757	rural	4	0.95	35	daylight	rainy	False	
4	517758	highway	2	0.86	35	daylight	clear	True	

In [24]:
```
# Convert the categorical columns into numerical
for col in test.columns:
    test[col]=label_encoder(test[col])
test_pred=tree.predict(test)
```

In [26]:
```
submission=pd.read_csv('/kaggle/input/playground-series-s5e10/sample_submission.csv')
submission['accident_risk']=test_pred
submission.to_csv('submission_2.csv',index=False)
```

In [ ]:
```

```