Start coding or generate with AI.

Samuel Maina Gachuru

Automobile Accident Severity Prediction according to weather condition.

Getting started.

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler  # Corrected class name
from sklearn.model_selection import train_test_split
import tensorflow as tf
```

```
df = pd.read_csv('https://raw.githubusercontent.com/mishrasarthak/Accident-Casualty-Prediction-System/refs/heads/master/Accident_
```

```
df
```

	Number of Vehicles	Road Surface	Lighting Conditions	Weather Conditions	Casualty Victim	Casualty Severity	Sex of Casualty	Age of Casualty	Type of Vehicle
0	5	3	1	4	1	1	1	36	6
1	5	3	1	4	2	1	1	27	6
2	1	2	2	1	3	2	1	68	4
3	2	1	1	1	1	1	1	49	4
4	2	2	1	1	1	1	1	33	4
...	...	...	...	...	...	...	...	...	...
2659	3	2	2	2	2	1	2	4	4
2660	2	1	2	1	1	1	2	23	4
2661	2	1	2	1	2	1	2	23	4
2662	1	1	2	1	3	1	2	76	4
2663	2	1	2	1	1	1	1	55	2

2664 rows × 9 columns

Data seperation into x and y

```
y = df['Weather Conditions']
y
```

```
      Weather Conditions
0                      4
1                      4
2                      1
3                      1
4                      1
...                  ...
2659                   2
2660                   1
2661                   1
2662                   1
2663                   1

2664 rows × 1 columns

dtype: int64
```

```
x = df.drop(['Weather Conditions', 'Number of Vehicles', 'Road Surface', 'Lighting Conditions', 'Casualty Victim', 'Sex of Casua
x
```

```
      Casualty Severity
0                     1
1                     1
2                     2
3                     1
4                     1
...                 ...
2659                  1
2660                  1
2661                  1
2662                  1
2663                  1

2664 rows × 1 columns
```

Data spliting into training and testing

```
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=100)
x_test
```

	Casualty Severity
2100	1
2559	1
2548	1
117	1
927	1
...	...
2385	1
42	1
685	2
1046	1
1645	1

533 rows × 1 columns

```
x_train
```

	Casualty Severity
474	1
1264	1
2552	1
1696	1
1671	2
...	...
350	1
1930	1
79	2
1859	2
1544	1

2131 rows × 1 columns

Model Building

Linear Regression

```
from sklearn.linear_model import LinearRegression
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=100)
```

```
model = LinearRegression()
```

Training The model

```
model.fit(X_train, y_train)
```

```
y_lr_test_pred = model.predict(X_test)
```

```
lr_test_mse = mean_squared_error(y_test, y_lr_test_pred)
lr_test_r2 = r2_score(y_test, y_lr_test_pred)
```

Applying the model to make prediction

```
from sklearn.metrics import mean_squared_error, r2_score
```

Evaluating The model performance

```
print(f"Test Mean Squared Error: {lr_test_mse}")
print(f"Test R-squared: {lr_test_r2}")
```

```
Test Mean Squared Error: 0.339310849105477
Test R-squared: -0.009007053107018104
```

```
lr_results = pd.DataFrame(['Linear Regression', lr_test_mse,lr_test_r2]).transpose()
lr_results.columns = ['method','TEST MSE','TEST R2']
lr_results
```

	method	TEST MSE	TEST R2
0	Linear Regression	0.339311	-0.009007

Data visualization

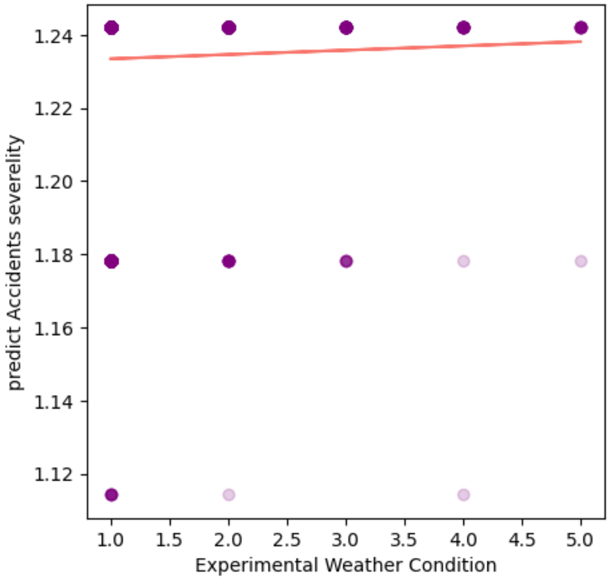
```
import matplotlib.pyplot as plt
import numpy as np

plt.figure(figsize=(5,5))
plt.scatter(x=y_train, y=y_lr_train_pred, c='purple', alpha=0.2)

z = np.polyfit(y_train, y_lr_train_pred, 1)
p = np.poly1d(z)


plt.plot(y_train, p(y_train), '#f8766D')
plt.ylabel('predict Accidents severelity')
plt.xlabel('Experimental Weather Condition')
```

```
Text(0.5, 0, 'Experimental Weather Condition')
```



```
Weather condition
```

Values Weather

1. Fine without high winds.
2. Raining without high winds
3. Raining with high winds.
4. [illegible]