Numerical data documentation

- Introduction
- Dataset
- Preprocessing
- Machine learning model
- Accuracy
- KNN

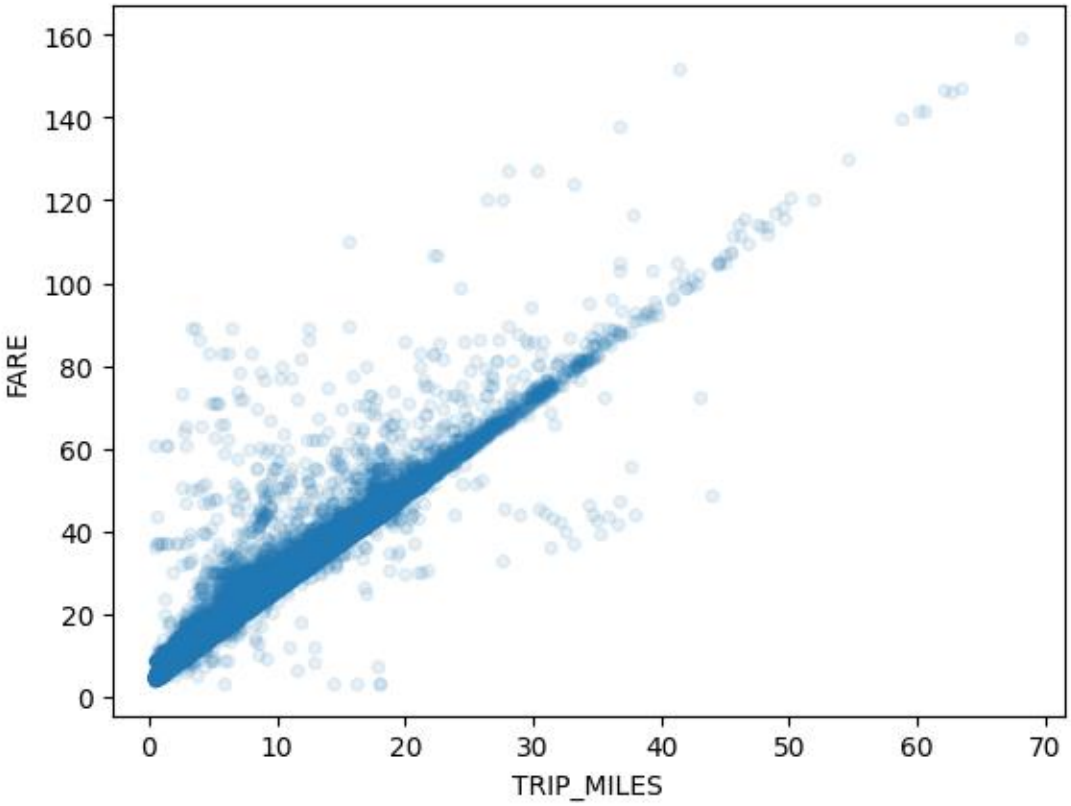
- **Introduction:**

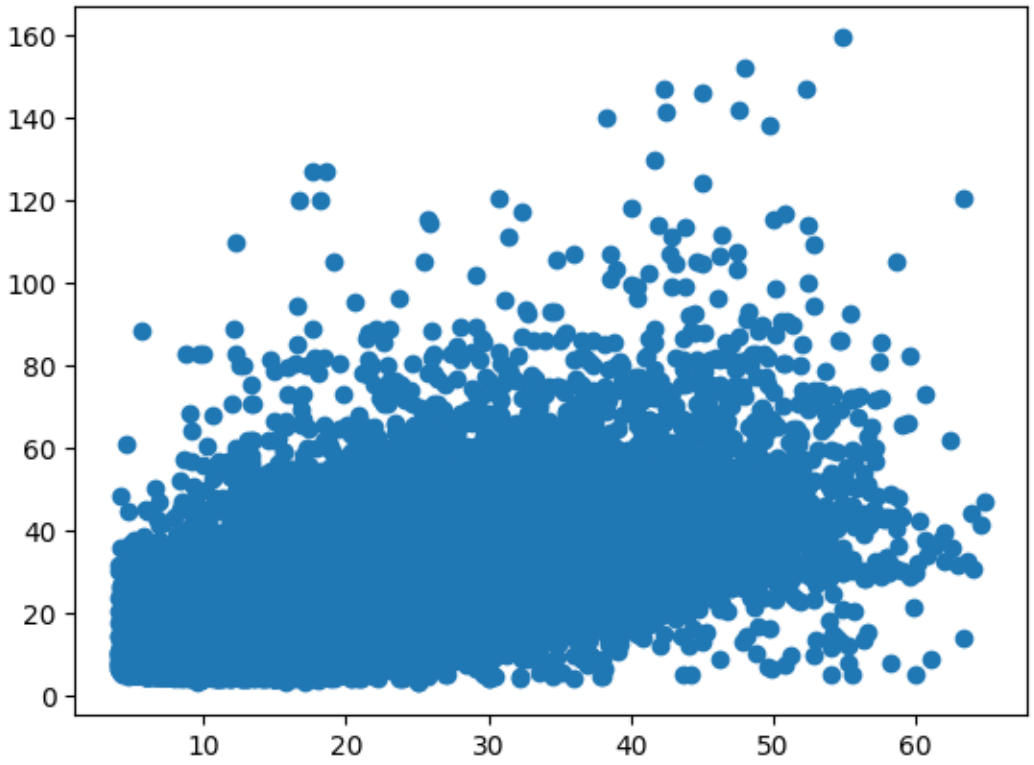
Predict the Fare of the taxi inside Chicago state

- **Dataset:**

‘Chicago taxi train’ csv file from (31695 X 18):
(TRIP_START_TIMESTAMP, TRIP_END_TIMESTAMP, TRIP_START_HOUR, TRIP_SECONDSTRIP_MILES, TRIP_SPEED, PICKUP_CENSUS_TRACT, DROPOFF_CENSUS_TRACT, PICKUP_COMMUNITY_AREA, DROPOFF_COMMUNITY_AREA, FARE, TIPS, TIP_RATE, TOLLS, EXTRAS, TRIP_TOTAL, PAYMENT_TYPE, COMPANY)
After creating a scatter matrix for ('TRIP_MILES','TRIP_SECONDS', 'TRIP_SPEED', 'FARE')

The most columns that effect the fair are Trip_miles and TRIP_SECONDS


160
140
120
100
80
60
40
20
0
FARE
0
10
20
30
40
50
60
70
TRIP_MILES


160
140
120
100
80
60
40
20
0
10
20
30
40
50
60

```
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   TRIP_START_TIMESTAMP    31694 non-null  object
 1   TRIP_END_TIMESTAMP      31694 non-null  float64
 2   TRIP_START_HOUR         31694 non-null  float64
 3   TRIP_SECONDS            31694 non-null  object
 4   TRIP_MILES              31661 non-null  float64
 5   TRIP_SPEED              31694 non-null  float64
 6   PICKUP_CENSUS_TRACT     13259 non-null  float64
 7   DROPOFF_CENSUS_TRACT    14023 non-null  float64
 8   PICKUP_COMMUNITY_AREA   28477 non-null  float64
 9   DROPOFF_COMMUNITY_AREA  28199 non-null  float64
 10  FARE                    31694 non-null  float64
 11  TIPS                    31694 non-null  float64
 12  TIP_RATE                31694 non-null  float64
 13  TOLLS                   31694 non-null  float64
 14  EXTRAS                  31694 non-null  float64
 15  TRIP_TOTAL              31694 non-null  float64
 16  PAYMENT_TYPE            31694 non-null  object
 17  COMPANY                 31694 non-null  object
dtypes: float64(14), object(4)
memory usage: 4.4+ MB
```

according to this table there are missing values in Trip_Miles and Trip_Speed cuz they are not equal to 31694

- **Preprocessing the data:**

Trip miles replacing the missing values with the mean

Trip seconds replacing the missing values with the mean

Using pandas library and replace function:

The code

```
df['TRIP_SECONDS']=df['TRIP_SECONDS'].replace("?",np.nan)

df['TRIP_SECONDS'] = pd.to_numeric(df['TRIP_SECONDS'])

mean_trip_seconds = df['TRIP_SECONDS'].mean()

df['TRIP_SECONDS']=df['TRIP_SECONDS'].fillna(mean_trip_seconds)

df.info()
```

- **Machine learning model:**

Now the data is ready to train the model using sklearn.linearRegression()

Starting with scalling the data due to the trip second has higher values than the trip mile so there isn't feature dominating other features.

The code:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(df[['TRIP_MILES','TRIP_SECONDS']])
```

then splitting the data into train and test:

```
from sklearn.model_selection import train_test_split

X = df[['TRIP_MILES','TRIP_SECONDS']]

y = df['FARE']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

The next step is to train the model:

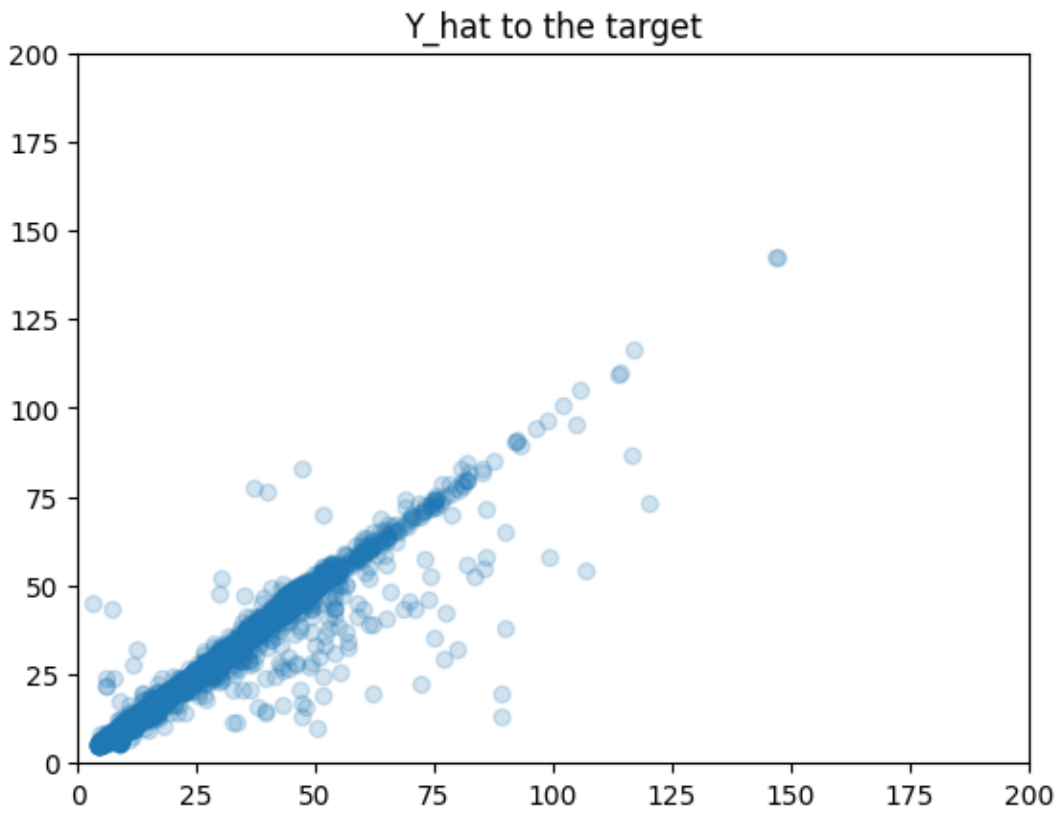
```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()

lin_reg.fit(X_train, y_train)

y_pred = lin_reg.predict(X_test)
```

y_train to Y_test graph:



The model perefore very will with the small data (0,30) after that it shows some outliers.

- **<u>Model accuracy:</u>**

Using mean squared error: a metric used to evaluate the performance of regression models. It calculates the average squared difference between the predicted and actual values.

The code:

```
from sklearn.metrics import mean_squared_error

lin_mse = mean_squared_error(y_test, y_pred)

lin_rmse = np.sqrt(lin_mse)
```

```
mean_actual = y_test.mean()

accuracy = 1 - (lin_rmse / mean_actual)

accuracy*100
```

the output is 84.8% its great number for this type of regression.

- **KNN: (K Neighbors Regressor):**

K-Nearest Neighbors (KNN) is a supervised machine learning algorithm used for classification and regression tasks. It identifies the *K* closest data points to make predictions.

```
from sklearn.neighbors import KNeighborsRegressor

knn = KNeighborsRegressor(n_neighbors=3)

knn.fit(X_train, y_train)

y_pred2 = knn.predict(X_test)

knn_mse = mean_squared_error(y_test, y_pred2)

knn_rmse = np.sqrt(knn_mse)

y_test_reset = y_test.reset_index(drop=True)

comparison_df = pd.DataFrame({
    'Actual': y_test_reset,
    'Predicted': y_pred2})
```

	Actual	Predicted
0	14.84	14.876667
1	4.75	6.166667
2	33.50	34.800000
3	23.00	23.583333
4	9.68	8.923333
...	...	...
6334	16.00	16.083333
6335	51.56	108.333333
6336	46.25	47.166667
6337	4.75	4.666667
6338	18.04	15.416667

[6339 rows x 2 columns]

MSE of KNN Regression model: 23.27

RMSE of KNN Regression model: 4.82