In [3]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge
import matplotlib.pyplot as plt
import kagglehub
import os
from sklearn.metrics import r2_score
```

In [4]:
```
path = kagglehub.dataset_download("chicago/chicago-taxi-rides-2016")
print(os.listdir(path))
```

```
Download already complete (493508776 bytes).
Extracting files...
['chicago_taxi_trips_2016_02.csv', 'column_remapping.json', 'data_dictionary.cs
v', 'chicago_taxi_trips_2016_11.csv', 'chicago_taxi_trips_2016_07.csv', 'chicag
o_taxi_trips_2016_12.csv', 'chicago_taxi_trips_2016_03.csv', 'chicago_taxi_trip
s_2016_08.csv', 'chicago_taxi_trips_2016_04.csv', 'chicago_taxi_trips_2016_05.c
sv', 'chicago_taxi_trips_2016_01.csv', 'chicago_taxi_trips_2016_09.csv', 'chica
go_taxi_trips_2016_10.csv', 'chicago_taxi_trips_2016_06.csv']
```

In [5]:
```
csv_path = os.path.join(path, 'chicago_taxi_trips_2016_10.csv')
df = pd.read_csv(csv_path)
df.head()
```

Out[5]:

	taxi_id	trip_start_timestamp	trip_end_timestamp	trip_seconds	trip_miles	p
0	2059.0	2016-10-2 16:45:00	2016-10-2 17:15:00	2460.0	17.4	
1	6308.0	2016-10-26 11:30:00	2016-10-26 12:00:00	1860.0	14.9	
2	2595.0	2016-10-21 12:00:00	2016-10-21 12:15:00	420.0	1.0	
3	6764.0	2016-10-21 23:15:00	2016-10-21 23:30:00	240.0	0.8	
4	3824.0	2016-10-2 00:00:00	2016-10-2 00:15:00	600.0	1.8	

In [6]:
```
df.shape
```

Out[6]:
```
(1499771, 20)
```

In [7]:
```
df.isnull().sum()
```

Out[7]:

	0
taxi_id	441
trip_start_timestamp	0
trip_end_timestamp	77
trip_seconds	88
trip_miles	40
pickup_census_tract	1499771
dropoff_census_tract	537186
pickup_community_area	139784
dropoff_community_area	170899
fare	43
tips	43
tolls	43
extras	43
trip_total	43
payment_type	0
company	634747
pickup_latitude	139743
pickup_longitude	139743
dropoff_latitude	167193
dropoff_longitude	167193

dtype: int64

In [10]:
```
numeric_cols = ['trip_seconds', 'trip_miles', 'fare', 'tips', 'tolls', 'extras
                'pickup_latitude', 'pickup_longitude', 'dropoff_latitude', 'dr
for col in numeric_cols:
    df[col].fillna(df[col].mean(), inplace=True)
```

```
/tmp/ipython-input-10-551096321.py:4: FutureWarning: A value is trying to be se
t on a copy of a DataFrame or Series through chained assignment using an inplac
e method.
The behavior will change in pandas 3.0. This inplace method will never work bec
ause the intermediate object on which we are setting values always behaves as a
copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.me
thod({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, t
o perform the operation inplace on the original object.


  df[col].fillna(df[col].mean(), inplace=True)
```

In [11]:
```
df.isnull().sum()
```

Out[11]:

	0
taxi_id	441
trip_start_timestamp	0
trip_end_timestamp	77
trip_seconds	0
trip_miles	0
pickup_census_tract	1499771
dropoff_census_tract	0
pickup_community_area	0
dropoff_community_area	0
fare	0
tips	0
tolls	0
extras	0
trip_total	0
payment_type	0
company	634747
pickup_latitude	0
pickup_longitude	0
dropoff_latitude	0
dropoff_longitude	0

dtype: int64

In [22]:
```
df = df.drop(["payment_type"], axis=1)
```

In [23]:
```
df.isnull().sum()
```

Out[23]:

	0
trip_seconds	0
trip_miles	0
dropoff_census_tract	0
pickup_community_area	0
dropoff_community_area	0
fare	0
tips	0
tolls	0
extras	0
trip_total	0
pickup_latitude	0
pickup_longitude	0
dropoff_latitude	0
dropoff_longitude	0

dtype: int64

In [24]:
```
x = df.drop("fare", axis=1).values
y = df["fare"].values
```

In [25]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=
```

In [26]:
```
model=Ridge()
model.fit(x_train,y_train)
prediction=model.predict(x_test)
print(prediction)
print(y_test)
```

```
[ 8.14512284  9.4567662  32.07549823 ...  8.64298083 11.69881978
 12.96292967]
[ 8.25  9.5  32.25 ...  8.75 11.75 13.  ]
```

In [27]:
```
r2 = r2_score(y_test, prediction)
print("R² Score:", r2)
```

```
R² Score: 0.9999161160821197
```

In [28]:
```
plt.scatter(y_test, prediction, color='blue')
```

```
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.title("Actual vs Predicted")
plt.show()
```