```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

file =
'https://raw.githubusercontent.com/amaydixit11/Academics/refs/heads/
main/DSL251/data.csv'
data = pd.read_csv(file)

data.head()
```

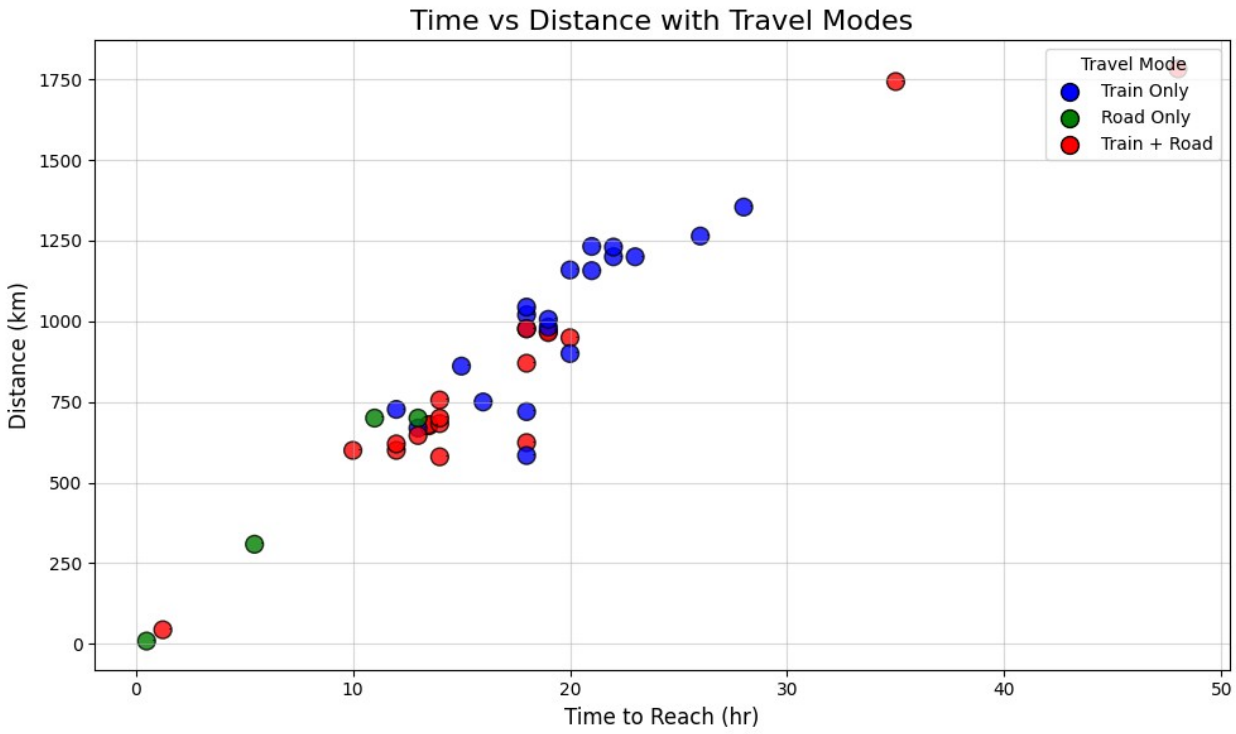

```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 43,\n  \"fields\":
[\n    {\n      \"column\": \"Location Name\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\": 35,\
n        \"samples\": [\n          \"Ongole \",\n
\"guntur\",\n          \"Varanasi\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Time to Reach (hr)\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
7.836545410100203,\n        \"min\": 0.5,\n        \"max\": 48.0,\n
\"num_unique_values\": 21,\n        \"samples\": [\n          18.0,\n
48.0,\n          10.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Distance (km)\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 354,\n        \"min\": 9,\n
\"max\": 1782,\n        \"num_unique_values\": 38,\n
\"samples\": [\n          900,\n          1159,\n          977\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Train Only\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          0,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Road Only\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Train+Road\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 0,\n
\"max\": 1,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          1,\n          0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"data"}
```


```
data.describe()
```

{"summary":"{\n \"name\": \"data\",\n \"rows\": 8,\n \"fields\": [\n   {\n   \"column\": \"Time to Reach (hr)\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 16.43339031746364,\n   \"min\": 0.5,\n   \"max\": 48.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   17.28418604651163,\n   18.0,\n   43.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Distance (km)\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 581.0372438911829,\n   \"min\": 9.0,\n   \"max\": 1782.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   865.5581395348837,\n   861.0,\n   43.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Train Only\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 15.058543923692291,\n   \"min\": 0.0,\n   \"max\": 43.0,\n   \"num_unique_values\": 5,\n   \"samples\": [\n   0.46511627906976744,\n   1.0,\n   0.5046845884077511\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Road Only\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 15.136616595487634,\n   \"min\": 0.0,\n   \"max\": 43.0,\n   \"num_unique_values\": 5,\n   \"samples\": [\n   0.09302325581395349,\n   1.0,\n   0.293902598732179\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Train+Road\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 15.058543923692291,\n   \"min\": 0.0,\n   \"max\": 43.0,\n   \"num_unique_values\": 5,\n   \"samples\": [\n   0.46511627906976744,\n   1.0,\n   0.5046845884077511\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
plt.figure(figsize=(10, 6))

plt.scatter(data["Time to Reach (hr)"], data["Distance (km)"],
c=colors, s=100, alpha=0.8, edgecolor="k")

for i, location in enumerate(data["Location Name"]):
    plt.text(data["Time to Reach (hr)"][i] + 0.2, data["Distance
(km)"][i], '.', fontsize=9)

plt.scatter([], [], color="blue", label="Train Only", s=100,
edgecolor="k")
plt.scatter([], [], color="green", label="Road Only", s=100,
edgecolor="k")
plt.scatter([], [], color="red", label="Train + Road", s=100,
edgecolor="k")

plt.title("Time vs Distance with Travel Modes", fontsize=16)
```

```
plt.xlabel("Time to Reach (hr)", fontsize=12)
plt.ylabel("Distance (km)", fontsize=12)
plt.legend(title="Travel Mode", loc="upper right")
plt.grid(alpha=0.5)

plt.tight_layout()
plt.show()
```




```
A = np.array([np.ones(data['Time to Reach (hr)'].shape), data['Time to Reach (hr)']])
A
```

```
array([[ 1.  ,  1.  ,  1.  ,  1.  ,  1.  ,  1.  ,  1.  ,  1.  ,  1.  ,
         1.  ,  1.  ,  1.  ,  1.  ,  1.  ,  1.  ,  1.  ,  1.  ,  1.  ,
         1.  ,  1.  ,  1.  ,  1.  ,  1.  ,  1.  ,  1.  ,  1.  ,  1.  ,
         1.  ,  1.  ,  1.  ,  1.  ,  1.  ,  1.  ,  1.  ,  1.  ,  1.  ,
         1.  ,  1.  ,  1.  ,  1.  ,  1.  ,  1.  ,  1.  ],
       [18.  , 18.  , 13.5 , 13.5 , 18.  , 14.  , 18.  , 21.  , 26.  ,
        15.  , 13.  ,  1.25,  0.5 , 14.  , 35.  , 19.  , 18.  , 12.  ,
        14.  , 22.  , 16.  , 21.  , 13.  ,  5.47, 10.  , 20.  , 12.  ,
        48.  , 11.  , 18.  , 12.  , 19.  , 19.  , 18.  , 28.  , 23.  ,
        19.  , 20.  , 13.  , 18.  , 14.  , 20.  , 22.  ]])
```

```
y = np.array(data['Distance (km)'])
y
```

```
array([1020,  870,  676,  680,  977,  580,  977, 1157, 1264,  861,
669,
         44,    9,  756, 1743,  970, 1044,  727,  683, 1200,  750,
1232,
        700,  309,  600,  949,  600, 1782,  700,  720,  620,  965,
982,
        624, 1354, 1200, 1006,  900,  646,  584,  700, 1159, 1230])
```

```
w_hat = np.linalg.inv(A.dot(A.T)).dot(A).dot(y)
w_hat
```

```
array([135.62292013,  42.23139102])
```

```
# verify that the error vector is perpendicular to any vector in
column space of A
e = (y - A.T.dot(w_hat))
e
```

```
array([ 124.2120415 ,  -25.7879585 ,  -29.74669891,  -25.74669891,
         81.2120415 , -146.86239442,   81.2120415 ,  134.51786844,
         30.36091333,   91.90621456,  -15.6310034 , -144.41215891,
       -147.73861564,   29.13760558,  129.27839415,   31.98065048,
        148.2120415 ,   84.60038762,  -43.86239442,  135.28647741,
        -61.32517646,  209.51786844,   15.3689966 ,  -57.62862901,
         42.06316966,  -31.25074054,  -42.39961238, -380.72968912,
         99.83177864, -175.7879585 ,  -22.39961238,   26.98065048,
         43.98065048, -271.7879585 ,   35.89813129,   93.05508639,
         67.98065048,  -80.25074054,  -38.6310034 , -311.7879585 ,
        -26.86239442,  178.74925946,  165.28647741])
```

```
data['error'] = e
data.describe()
```

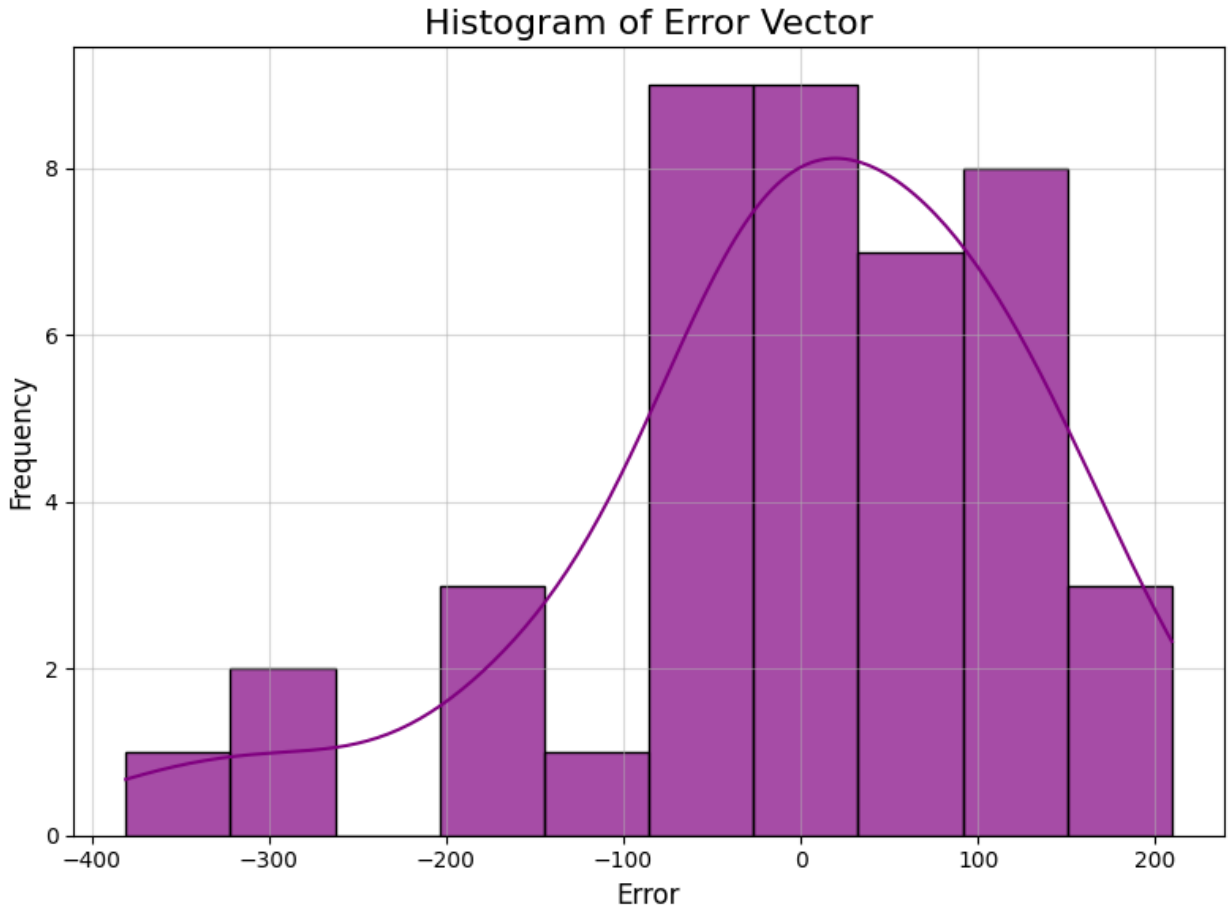
```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 8,\n  \"fields\": [\
n    {\n      \"column\": \"Time to Reach (hr)\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
16.43339031746364,\n        \"min\": 0.5,\n        \"max\": 48.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
17.28418604651163,\n          18.0,\n          43.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Distance (km)\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
581.0372438911829,\n        \"min\": 9.0,\n        \"max\": 1782.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
865.5581395348837,\n          861.0,\n          43.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Train Only\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
15.058543923692291,\n        \"min\": 0.0,\n        \"max\": 43.0,\n
```

\"num_unique_values\": 5,\n   \"samples\": [\n
0.46511627906976744,\n   1.0,\n   0.5046845884077511\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"Road Only\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
15.136616595487634,\n   \"min\": 0.0,\n   \"max\": 43.0,\n
\"num_unique_values\": 5,\n   \"samples\": [\n
0.09302325581395349,\n   1.0,\n   0.293902598732179\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"Train+Road\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
15.058543923692291,\n   \"min\": 0.0,\n   \"max\": 43.0,\n
\"num_unique_values\": 5,\n   \"samples\": [\n
0.46511627906976744,\n   1.0,\n   0.5046845884077511\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"error\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\":
176.00203165146118,\n   \"min\": -380.72968912285705,\n
\"max\": 209.5178684352809,\n   \"num_unique_values\": 8,\n
\"samples\": [\n   3.2784111342977643e-13,\n
26.98065047662442,\n   43.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe"}

```
# error is perpendicular to A
A.dot(e)
```

```
array([1.42108547e-11, 3.01952241e-10])
```

```
plt.figure(figsize=(8, 6))
sns.histplot(data["error"], kde=True, bins=10, color="purple",
alpha=0.7, edgecolor="black")

plt.title("Histogram of Error Vector", fontsize=16)
plt.xlabel("Error", fontsize=12)
plt.ylabel("Frequency", fontsize=12)
plt.grid(alpha=0.5)

plt.tight_layout()
plt.show()
```




```
road_only = data[data['Road Only'] == 1]
road_only_A = np.array([np.ones(road_only['Time to Reach
(hr)'].shape), road_only['Time to Reach (hr)']])
road_only_y = np.array(road_only['Distance (km)'])
road_only_w_hat =
np.linalg.inv(road_only_A.dot(road_only_A.T)).dot(road_only_A).dot(roa
d_only_y)
road_only_w_hat
```

```
array([-11.06834871,  58.80124774])
```

```
road_only_e = (road_only_y - road_only_A.T.dot(road_only_w_hat))
road_only_e
```

```
array([ -9.33227516, -53.34787194,  -1.57447644,  64.25462354])
```

```
road_only['road_only_error'] = road_only_e
```

```
<ipython-input-136-4733fb8f6d1c>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  road_only['road_only_error'] = road_only_e
```

```
road_only.describe()
```


```
{"summary":"{\n  \"name\": \"road_only\",\n  \"rows\": 8,\n
\"fields\": [\n    {\n      \"column\": \"Time to Reach (hr)\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
4.115198107636982,\n        \"min\": 0.5,\n        \"max\": 13.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n          7.4925,\n
8.235,\n          4.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Distance (km)\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 273.3802196657122,\n        \"min\":
4.0,\n        \"max\": 700.0,\n        \"num_unique_values\": 7,\n
\"samples\": [\n          4.0,\n          429.5,\n          504.5\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Train Only\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1.4142135623730951,\n        \"min\": 0.0,\n        \"max\": 4.0,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          0.0,\n
4.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Road Only\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1.164964745021435,\n        \"min\":
0.0,\n        \"max\": 4.0,\n        \"num_unique_values\": 3,\n
\"samples\": [\n          4.0,\n          1.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Train+Road\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1.4142135623730951,\n        \"min\": 0.0,\n        \"max\": 4.0,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          0.0,\n
4.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"error\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 85.80887311510278,\n        \"min\": -147.73861564094767,\n
\"max\": 105.38477035018127,\n        \"num_unique_values\": 8,\n
\"samples\": [\n          -22.541617352995075,\n          -
21.129816206515642\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"road_only_error\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 37.15858218575386,\n        \"min\": -
53.34787194296655,\n        \"max\": 64.25462354245019,\n
\"num_unique_values\": 8,\n        \"samples\": [\n          -
1.7763568394002505e-13,\n          -5.453375799742176\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```