RIASEC ANALYSIS

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
```

In [2]:
```
#df = pd.read_csv("RIASEC.csv")
```

Reading and Cleaning

In [3]:
```
df = pd.read_csv("RIASEC.csv", sep=None, engine="python")

print(df.shape)
print(df.head())
```

```
(8855, 55)
   implementation  R1  R2  R3  R4  R5  R6  R7  R8  I1  ...  C5  C6  C7  C8  \
0               2   3   1   4   2   1   2   1   1   5  ...   2   1   1   2
1               2   1   1   1   1   1   1   1   1   4  ...   1   1   1   1
2               2   3   2   1   1   1   1   2   1   5  ...   3   4   4   4
3               2   3   2   1   2   2   3   1   2   5  ...   1   3   2   1
4               2  -1   2   3   2   3   2   1   3   5  ...   4   3   3   3

   accuracy  elapse  country  fromsearch  age  gender
0        90     222       PT           0   -1      -1
1       100     102       US           0   -1      -1
2        95     264       US           1   -1      -1
3        60     189       SG           0   -1      -1
4        90     197       US           0   -1      -1

[5 rows x 55 columns]
```

In [4]:
```
cols = [f"R{i}" for i in range(1, 9)]
print(cols)
R_data = df[cols]
R_data.shape
```

```
['R1', 'R2', 'R3', 'R4', 'R5', 'R6', 'R7', 'R8']
```

Out[4]:
```
(8855, 8)
```

In [5]:
```
df = R_data[(R_data != -1).all(axis=1)] # cleans the data and gets rid of rows w
print("Shape after cleaning:", df.shape)
```

```
Shape after cleaning: (8478, 8)
```

In [6]:
```
# the entries have reduced from 8855 people to 8478 meaning 377 people had inval
```

In [7]:
```
8855-8478
```

Out[7]:
```
377
```

In [8]:
```
df
```

Out[8]:

	R1	R2	R3	R4	R5	R6	R7	R8
0	3	1	4	2	1	2	1	1
1	1	1	1	1	1	1	1	1
2	3	2	1	1	1	1	2	1
3	3	2	1	2	2	3	1	2
5	3	1	3	4	3	4	3	3
...	...	...	...	...	...	...	...	...
8849	3	3	1	4	2	2	2	1
8851	3	2	3	4	3	2	2	2
8852	4	3	3	3	2	2	1	2
8853	4	4	3	5	4	5	3	4
8854	4	2	4	4	1	4	2	3

8478 rows × 8 columns

In [9]:

```
df["R1"]
```

Out[9]:

```
0       3
1       1
2       3
3       3
5       3
       ..
8849    3
8851    3
8852    4
8853    4
8854    4
Name: R1, Length: 8478, dtype: int64
```

Model selection

In [40]:

```
df = df.copy()
df["R_score"] = df.mean(axis=1) #row wise mean
df.shape
```

Out[40]:

```
(8478, 9)
```

In [41]:

```
train = df.iloc[:6500]
test = df.iloc[6500:]
train.shape, test.shape
```

Out[41]:

```
((6500, 9), (1978, 9))
```

In [42]:

```
x_train = train[["R1"]]
y_train = train["R_score"]
```

In [43]:
```
model = LinearRegression()
model.fit(x_train, y_train)
```

Out[43]:
```
▾ LinearRegression

LinearRegression()
```

In [44]:
```
print("Intercept:", model.intercept_)
print("Coefficient for R1:", model.coef_[0])

model.coef_.shape
```

```
Intercept: 1.018105231431332
Coefficient for R1: 0.42359934763433976
```

Out[44]:
```
(1,)
```

Hence the form of the best-fit regression line is $R_{score} = R_1 \times \text{coef} + \text{intercept}$

In [45]:
```
y_pred_train = model.predict(x_train)
RSS_train = np.sum((y_train - y_pred_train) ** 2) #RSS -> residual sum of square
RSS_train_avg = RSS_train / len(train)
print("Training RSS (total):", RSS_train)
print("Training RSS (average):", RSS_train_avg)
```

```
Training RSS (total): 2902.0393474685015
Training RSS (average): 0.446467591918231
```

Validation

In [46]:
```
x_test = test[["R1"]]
y_test = test["R_score"]
```

In [47]:
```
y_pred_test = model.predict(x_test)
RSS_test = np.sum((y_test - y_pred_test) ** 2)
RSS_test_avg = RSS_test / len(test)
```

In [48]:
```
print("Test RSS (total):", RSS_test)
print("Test RSS (average):", RSS_test_avg)
```

```
Test RSS (total): 1028.7757850021012
Test RSS (average): 0.5201090925187569
```

In [49]:
```
print(f"Average RSS (train): {RSS_train_avg:.5f}")
print(f"Average RSS (test): {RSS_test_avg:.5f}")
```

```
Average RSS (train): 0.44647
Average RSS (test): 0.52011
```

Model works better with training data

For other fields

Training

In [50]:
```
models = {}
RSS_train = {}
RSS_train_avg = {}

for i in range(2, 9):   # R2 to R8
    x_train = train[[f"R{i}"]]
    y_train = train["R_score"]

    models[i] = LinearRegression()
    models[i].fit(x_train, y_train)

    y_pred_train = models[i].predict(x_train)

    RSS_train[i] = np.sum((y_train - y_pred_train) ** 2)
    RSS_train_avg[i] = RSS_train[i] / len(train)

    print(f"R{i}:")
    print(f"  Training RSS (total): {RSS_train[i]}")
    print(f"  Training RSS (average): {RSS_train_avg[i]}\n")
```

```
R2:
  Training RSS (total): 2086.9934694791536
  Training RSS (average): 0.32107591838140825

R3:
  Training RSS (total): 2856.1295910903655
  Training RSS (average): 0.43940455247544086

R4:
  Training RSS (total): 2058.2069482183156
  Training RSS (average): 0.31664722280281776

R5:
  Training RSS (total): 2026.8220877210033
  Training RSS (average): 0.3118187827263082

R6:
  Training RSS (total): 1762.7887420860695
  Training RSS (average): 0.2711982680132415

R7:
  Training RSS (total): 1928.2557723088821
  Training RSS (average): 0.29665473420136645

R8:
  Training RSS (total): 1771.9059588635785
  Training RSS (average): 0.2726009167482428
```

In [51]:
```
dict(sorted(RSS_train_avg.items(), key=lambda item: item[1]))
```

Out[51]:
```
{6: np.float64(0.2711982680132415),
 8: np.float64(0.2726009167482428),
 7: np.float64(0.29665473420136645),
 5: np.float64(0.3118187827263082),
 4: np.float64(0.31664722280281776),
 2: np.float64(0.32107591838140825),
 3: np.float64(0.43940455247544086)}
```

In [52]:
```
keys = list(RSS_train_avg.keys())
values = list(RSS_train_avg.values())
best_feature = keys[np.argmin(values)]
print(f"Best feature is {best_feature} based on training data")
```

```
Best feature is 6 based on training data
```

Testing

In [53]:
```
models = {}
RSS_test = {}
RSS_test_avg = {}

for i in range(2, 9):   # R2 to R8
    x_test = test[[f"R{i}"]]
    y_test = test["R_score"]

    models[i] = LinearRegression()
    models[i].fit(x_test, y_test)

    y_pred_test = models[i].predict(x_test)

    RSS_test[i] = np.sum((y_test - y_pred_test) ** 2)
    RSS_test_avg[i] = RSS_test[i] / len(test)

    print(f"R{i}:")
    print(f"  Testing RSS (total): {RSS_test[i]}")
    print(f"  Testing RSS (average): {RSS_test_avg[i]}\n")
```

```
R2:
  Testing RSS (total): 712.0378480768431
  Testing RSS (average): 0.35997868962428875

R3:
  Testing RSS (total): 875.312355121421
  Testing RSS (average): 0.4425239409107285

R4:
  Testing RSS (total): 799.8393275744875
  Testing RSS (average): 0.4043677085816418

R5:
  Testing RSS (total): 697.1299733893734
  Testing RSS (average): 0.3524418470118167

R6:
  Testing RSS (total): 630.9467824594283
  Testing RSS (average): 0.31898219537888184

R7:
  Testing RSS (total): 680.1640236301748
  Testing RSS (average): 0.3438645215521612

R8:
  Testing RSS (total): 678.3905837840575
  Testing RSS (average): 0.3429679392234871
```

In [54]:
```
dict(sorted(RSS_test_avg.items(), key=lambda item: item[1]))
```

Out[54]:
```
{6: np.float64(0.31898219537888184),
 8: np.float64(0.3429679392234871),
 7: np.float64(0.3438645215521612),
 5: np.float64(0.3524418470118167),
 2: np.float64(0.35997868962428875),
 4: np.float64(0.4043677085816418),
 3: np.float64(0.4425239409107285)}
```

In [55]:
```
keys = list(RSS_test_avg.keys())
values = list(RSS_test_avg.values())
best_feature = keys[np.argmin(values)]
print(f"Best feature is {best_feature} based on testing data")
```

```
Best feature is 6 based on testing data
```

We can collectively agree that for mean value based regression, 6 is the best performing fitting parameter.

In [ ]: