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

```
path = "//content//drive//MyDrive//ML_testFile//study_performance.csv"
df = pd.read_csv(path)
df.head(40)
```

	gender	race_ethnicity	parental_level_of_education	lunch	test_preparation_course	math_score	reading_score	writing_score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75
5	female	group B	associate's degree	standard	none	71	83	78
6	female	group B	some college	standard	completed	88	95	92
7	male	group B	some college	free/reduced	none	40	43	39
8	male	group D	high school	free/reduced	completed	64	64	67
9	female	group B	high school	free/reduced	none	38	60	50
10	male	group C	associate's degree	standard	none	58	54	52
11	male	group D	associate's degree	standard	none	40	52	43
12	female	group B	high school	standard	none	65	81	73
13	male	group A	some college	standard	completed	78	72	70
14	female	group A	master's degree	standard	none	50	53	58
15	female	group C	some high school	standard	none	69	75	78
16	male	group C	high school	standard	none	88	89	86
17	female	group B	some high school	free/reduced	none	18	32	28
18	male	group C	master's degree	free/reduced	completed	46	42	46
19	female	group C	associate's degree	free/reduced	none	54	58	61
20	male	group D	high school	standard	none	66	69	63
21	female	group B	some college	free/reduced	completed	65	75	70
22	male	group D	some college	standard	none	44	54	53
23	female	group C	some high school	standard	none	69	73	73
24	male	group D	bachelor's degree	free/reduced	completed	74	71	80
25	male	group A	master's degree	free/reduced	none	73	74	72
26	male	group B	some college	standard	none	69	54	55
27	female	group C	bachelor's degree	standard	none	67	69	75
28	male	group C	high school	standard	none	70	70	65
29	female	group D	master's degree	standard	none	62	70	75
30	female	group D	some college	standard	none	69	74	74
31	female	group B	some college	standard	none	63	65	61
32	female	group E	master's degree	free/reduced	none	56	72	65
33	male	group D	some college	standard	none	40	42	38
34	male	group E	some college	standard	none	97	87	82
35	male	group E	associate's degree	standard	completed	81	81	79
36	female	group D	associate's degree	standard	none	74	81	83
37	female	group D	some high school	free/reduced	none	50	64	59
38	female	group D	associate's degree	free/reduced	completed	75	90	88
39	male	group B	associate's degree	free/reduced	none	57	56	57

```
df["mean_score"] = ((df["math_score"] + df["reading_score"] + df["writing_score"]) / 3).round()
df["mean_score"]
```

```
0      73.0
1      82.0
2      93.0
3      49.0
4      76.0
```

```
      ...
995    94.0
996    57.0
997    65.0
998    74.0
999    83.0
Name: mean_score, Length: 1000, dtype: float64
```

```
from sklearn.preprocessing import LabelEncoder
lc = LabelEncoder()
df['gender'] = lc.fit_transform(df['gender'])
df['race_ethnicity'] = lc.fit_transform(df['race_ethnicity'])
df['parental_level_of_education'] = lc.fit_transform(df['parental_level_of_education'])
df['lunch'] = lc.fit_transform(df['lunch'])
df['test_preparation_course'] = lc.fit_transform(df['test_preparation_course'])
df.head(40)
```

	gender	race_ethnicity	parental_level_of_education	lunch	test_preparation_course	m
0	0	1	1	1	1	
1	0	2	4	1	0	
2	0	1	3	1	1	
3	1	0	0	0	1	
4	1	2	4	1	1	
5	0	1	0	1	1	
6	0	1	4	1	0	
7	1	1	4	0	1	
8	1	3	2	0	0	
9	0	1	2	0	1	
10	1	2	0	1	1	
11	1	3	0	1	1	
12	0	1	2	1	1	
13	1	0	4	1	0	
14	0	0	3	1	1	
15	0	2	5	1	1	
16	1	2	2	1	1	
17	0	1	5	0	1	
18	1	2	3	0	0	
19	0	2	0	0	1	
20	1	3	2	1	1	
21	0	1	4	0	0	
22	1	3	4	1	1	
23	0	2	5	1	1	
24	1	3	1	0	0	
25	1	0	3	0	1	
26	1	1	4	1	1	
27	0	2	1	1	1	
28	1	2	2	1	1	
29	0	3	3	1	1	
30	0	3	4	1	1	
31	0	1	4	1	1	
32	0	4	3	0	1	
33	1	3	4	1	1	
34	1	4	4	1	1	
35	1	4	0	1	0	
36	0	3	0	1	1	
37	0	3	5	0	1	
38	0	3	0	0	0	
39	1	1	0	0	1	

```
df = df.drop(['math_score', 'writing_score', 'reading_score'], axis=1)
```

```
from sklearn.model_selection import train_test_split
y = df['mean_score']
X = df.drop(['mean_score'], axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=51)
X_train
```

	gender	race_ethnicity	parental_level_of_education	lunch	test_preparation_course
618	1	3	3	1	1
324	0	2	2	0	1
740	1	3	1	1	1
78	0	3	5	1	0
665	0	2	5	0	0
...	...	...	...	...	...
709	0	3	0	0	0
969	0	1	1	1	1
736	1	2	0	1	1
485	1	2	2	1	1
57	1	3	0	1	1

800 rows × 5 columns

```
X_test
```

	gender	race_ethnicity	parental_level_of_education	lunch	test_preparation_course
833	0	1	2	1	0
551	1	1	1	0	0
782	0	1	2	0	0
535	0	2	1	0	0
144	1	3	4	1	1
...	...	...	...	...	...
975	0	2	4	1	0
971	1	2	5	1	0
366	1	2	2	1	0
479	1	4	0	1	1
870	1	1	2	1	1

200 rows × 5 columns

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train, y_train)
y_lr_train_pred = lr.predict(X_train)
y_lr_test_pred = lr.predict(X_test)
```

```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(max_depth=2, random_state=0)
rf.fit(X_train, y_train)
y_rf_train_pred = rf.predict(X_train)
y_rf_test_pred = rf.predict(X_test)
```

```
from sklearn.metrics import mean_absolute_error, r2_score
lr_train_mae = mean_absolute_error(y_train, y_lr_train_pred)
lr_train_r2 = r2_score(y_train, y_lr_train_pred)
lr_test_mae = mean_absolute_error(y_test, y_lr_test_pred)
lr_test_r2 = r2_score(y_test, y_lr_test_pred)

rf_train_mae = mean_absolute_error(y_train, y_rf_train_pred)
rf_train_r2 = r2_score(y_train, y_rf_train_pred)
rf_test_mae = mean_absolute_error(y_test, y_rf_test_pred)
rf_test_r2 = r2_score(y_test, y_rf_test_pred)
```

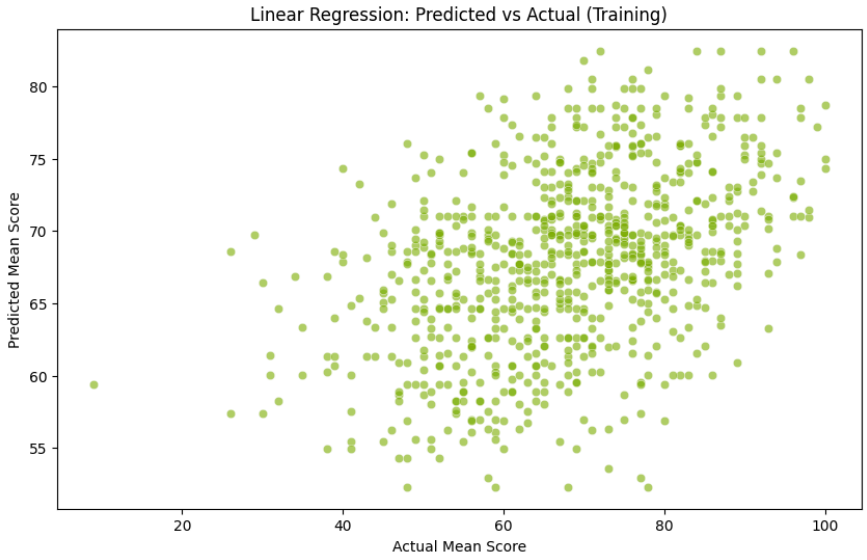
```
from sklearn.neighbors import KNeighborsRegressor
knn = KNeighborsRegressor(n_neighbors=5)
knn.fit(X_train, y_train)
y_knn_train_pred = knn.predict(X_train)
y_knn_test_pred = knn.predict(X_test)
knn_train_mae = mean_absolute_error(y_train, y_knn_train_pred)
knn_train_r2 = r2_score(y_train, y_knn_train_pred)
knn_test_mae = mean_absolute_error(y_test, y_knn_test_pred)
knn_test_r2 = r2_score(y_test, y_knn_test_pred)
```

```
lr_results = pd.DataFrame(['Linear Regression', lr_train_mae, lr_train_r2, lr_test_mae, lr_test_r2]).transpose()
lr_results.columns = ['Method', 'Training MAE', 'Training R2', 'Test MAE', 'Test R2']

rf_results = pd.DataFrame(['Random Forest', rf_train_mae, rf_train_r2, rf_test_mae, rf_test_r2]).transpose()
rf_results.columns = ['Method', 'Training MAE', 'Training R2', 'Test MAE', 'Test R2']

knn_results = pd.DataFrame(['K-Nearest Neighbors', knn_train_mae, knn_train_r2, knn_test_mae, knn_test_r2]).transpose()
knn_results.columns = ['Method', 'Training MAE', 'Training R2', 'Test MAE', 'Test R2']
```

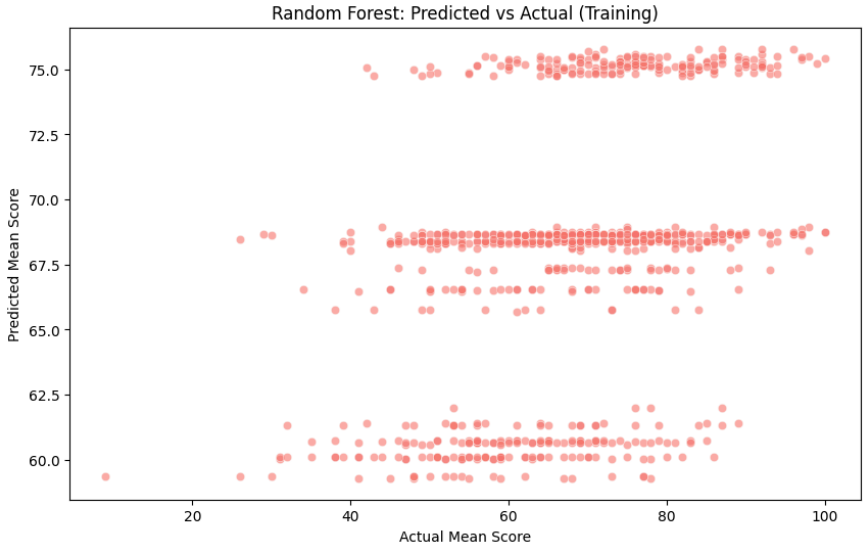
```
df_models = pd.concat([lr_results, rf_results, knn_results], axis=0).reset_index(drop=True)
print(df_models)
```

```
                  Method Training MAE Training R2   Test MAE   Test R2
0    Linear Regression    10.169091    0.189306   10.07442  0.263734
1        Random Forest    10.378422    0.154793  10.469843  0.189953
2  K-Nearest Neighbors       9.6965    0.279106     10.886   0.09852
```

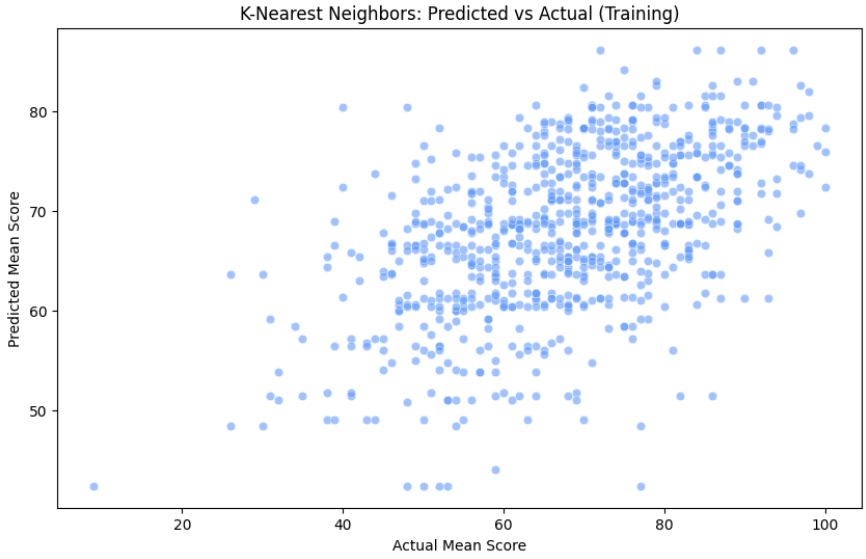
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=y_train, y=y_lr_train_pred, color="#7CAE00", alpha=0.6)
plt.title('Linear Regression: Predicted vs Actual (Training)')
plt.xlabel('Actual Mean Score')
plt.ylabel('Predicted Mean Score')
plt.show()
```



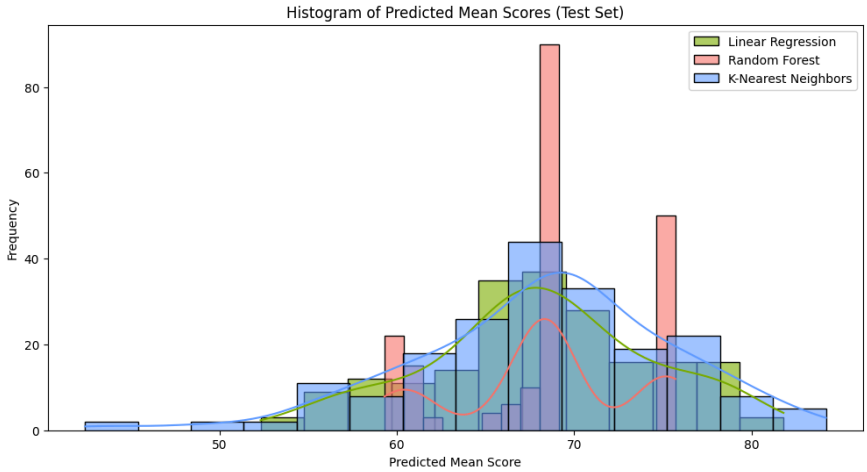

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=y_train, y=y_rf_train_pred, color="#F8766D", alpha=0.6)
plt.title('Random Forest: Predicted vs Actual (Training)')
plt.xlabel('Actual Mean Score')
plt.ylabel('Predicted Mean Score')
plt.show()
```




```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=y_train, y=y_knn_train_pred, color="#619CFF", alpha=0.6)
plt.title('K-Nearest Neighbors: Predicted vs Actual (Training)')
plt.xlabel('Actual Mean Score')
plt.ylabel('Predicted Mean Score')
plt.show()
```




```
plt.figure(figsize=(12, 6))
sns.histplot(y_lr_test_pred, color="#7CAE00", label='Linear Regression', alpha=0.6, kde=True)
sns.histplot(y_rf_test_pred, color="#F8766D", label='Random Forest', alpha=0.6, kde=True)
sns.histplot(y_knn_test_pred, color="#619CFF", label='K-Nearest Neighbors', alpha=0.6, kde=True)
plt.title('Histogram of Predicted Mean Scores (Test Set)')
plt.xlabel('Predicted Mean Score')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```




```
print("Linear Regression Test MAE:", lr_test_mae)
print("Random Forest Test MAE:", rf_test_mae)
print("K-Nearest Neighbors Test MAE:", knn_test_mae)
```

```
Linear Regression Test MAE: 10.074419910948786
Random Forest Test MAE: 10.469842639703401
K-Nearest Neighbors Test MAE: 10.886
```

```
print("\nLinear Regression Test R2 Score:", lr_test_r2)
print("Random Forest Test R2 Score:", rf_test_r2)
print("K-Nearest Neighbors Test R2 Score:", knn_test_r2)
```

```
Linear Regression Test R2 Score: 0.26373392207493596
Random Forest Test R2 Score: 0.18995269310323082
K-Nearest Neighbors Test R2 Score: 0.09851964306631511
```

```
import joblib
joblib.dump(lr, "students_performance_predictor.pkl")
```

```
['students_performance_predictor.pkl']
```

```
model = joblib.load("students_performance_predictor.pkl")
```

```
model.predict([[0, 4, 2, 1, 0]])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression was fit
  warnings.warn(
array([81.15185469])
```