In [49]:
```
import pandas as pd
import numpy as np
import math
import matplotlib.pyplot as plt
import statistics as stats
```

In [51]:
```
data=pd.read_csv('big4_financial_risk_compliance.csv' , header='infer')
print("Data info :- \n",data.info(),'\n')
print('Number of different companise in dataset :- ',np.unique(data.loc[:,'Firm_Name']))
print('No. of rows and columns :-\n',data.shape,'\n')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Year                       100 non-null    int64  
 1   Firm_Name                  100 non-null    object 
 2   Total_Audit_Engagements    100 non-null    int64  
 3   High_Risk_Cases            100 non-null    int64  
 4   Compliance_Violations      100 non-null    int64  
 5   Fraud_Cases_Detected       98 non-null     float64
 6   Industry_Affected          100 non-null    object 
 7   Total_Revenue_Impact       100 non-null    float64
 8   AI_Used_for_Auditing       100 non-null    object 
 9   Employee_Workload          100 non-null    int64  
 10  Audit_Effectiveness_Score  100 non-null    float64
 11  Client_Satisfaction_Score  100 non-null    float64
dtypes: float64(4), int64(5), object(3)
memory usage: 9.5+ KB
Data info :-
 None

Number of different companise in dataset :-  ['Deloitte' 'Ernst & Young' 'KPMG' 'PwC']
No. of rows and columns :-
 (100, 12)
```

In [53]:
```
#Checking null vvalues
d1=data.copy()
d1.dropna(axis=1)
d1.info()
d1.shape
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Year                       100 non-null    int64  
 1   Firm_Name                  100 non-null    object 
 2   Total_Audit_Engagements    100 non-null    int64  
 3   High_Risk_Cases            100 non-null    int64  
 4   Compliance_Violations      100 non-null    int64  
 5   Fraud_Cases_Detected       98 non-null     float64
 6   Industry_Affected          100 non-null    object 
 7   Total_Revenue_Impact       100 non-null    float64
 8   AI_Used_for_Auditing       100 non-null    object 
 9   Employee_Workload          100 non-null    int64  
 10  Audit_Effectiveness_Score  100 non-null    float64
 11  Client_Satisfaction_Score  100 non-null    float64
dtypes: float64(4), int64(5), object(3)
memory usage: 9.5+ KB
```

Out[53]:
```
(100, 12)
```

In [57]:
```
#Handling missing values
d2=data.copy()
Fraud_Cases_Detected_mean = np.ceil(np.mean(d2.loc[ ~d2.loc[:,'Fraud_Cases_Detected'].isna(),'Fraud_Cases_Detec
Client_Satisfaction_Score_mean = np.mean(d2.loc[ ~d2.loc[:,'Client_Satisfaction_Score'].isna(),'Client_Satisfac
AI_Used_for_Auditing_mode = stats.mode(d2.loc[ ~d2.loc[:,'AI_Used_for_Auditing'].isna(),'AI_Used_for_Auditing']

print('Fraud_Cases_Detected =',Fraud_Cases_Detected_mean)
print('Client_Satisfaction_Score =',Client_Satisfaction_Score_mean)
print('AI_Used_for_Auditing =',AI_Used_for_Auditing_mode)

d2.loc[d2.loc[:,'Fraud_Cases_Detected'].isna(),'Fraud_Cases_Detected'] = Fraud_Cases_Detected_mean
d2.loc[d2.loc[:,'Client_Satisfaction_Score'].isna(),'Client_Satisfaction_Score'] = Client_Satisfaction_Score_mea
d2.loc[d2.loc[:,'AI_Used_for_Auditing'].isna(),'AI_Used_for_Auditing'] = AI_Used_for_Auditing_mode[0]

d2.info()
```

```
Fraud_Cases_Detected = 53.0
Client_Satisfaction_Score = 7.338999999999999
AI_Used_for_Auditing = No
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Year                       100 non-null    int64
 1   Firm_Name                  100 non-null    object
 2   Total_Audit_Engagements    100 non-null    int64
 3   High_Risk_Cases            100 non-null    int64
 4   Compliance_Violations      100 non-null    int64
 5   Fraud_Cases_Detected       100 non-null    float64
 6   Industry_Affected          100 non-null    object
 7   Total_Revenue_Impact       100 non-null    float64
 8   AI_Used_for_Auditing       100 non-null    object
 9   Employee_Workload          100 non-null    int64
 10  Audit_Effectiveness_Score  100 non-null    float64
 11  Client_Satisfaction_Score  100 non-null    float64
dtypes: float64(4), int64(5), object(3)
memory usage: 9.5+ KB
```

In [35]:

```python
d3=data.copy()

print('Statistical analysis of column Total_Audit_Engagements :-\n')
sum_Total_Audit_Engagements = np.sum(data.loc[:,'Total_Audit_Engagements'])
mean_Total_Audit_Engagements = np.mean(data.loc[:,'Total_Audit_Engagements'])
median_Total_Audit_Engagements = np.median(data.loc[:,'Total_Audit_Engagements'])
variance_Total_Audit_Engagements = np.var(data.loc[:,'Total_Audit_Engagements'])
standard_deviation_Total_Audit_Engagements = np.std(data.loc[:,'Total_Audit_Engagements'])
min_Total_Audit_Engagements = np.min(data.loc[:,'Total_Audit_Engagements'])
max_Total_Audit_Engagements = np.max(data.loc[:,'Total_Audit_Engagements'])
print('Sum =',sum_Total_Audit_Engagements)
print('Range :- [',min_Total_Audit_Engagements,',',max_Total_Audit_Engagements,']')
print('Min =',min_Total_Audit_Engagements)
print('Max =',max_Total_Audit_Engagements)
print('Mean =',mean_Total_Audit_Engagements)
print('Median =',median_Total_Audit_Engagements)
print('Variance =',variance_Total_Audit_Engagements)
print('Standard Deviation =',standard_deviation_Total_Audit_Engagements,'\n\n')


print('Statistical analysis of column High_Risk_Cases :-\n')
sum_High_Risk_Cases = np.sum(data.loc[:,'High_Risk_Cases'])
mean_High_Risk_Cases = np.mean(data.loc[:,'High_Risk_Cases'])
median_High_Risk_Cases = np.median(data.loc[:,'High_Risk_Cases'])
variance_High_Risk_Cases = np.var(data.loc[:,'High_Risk_Cases'])
standard_deviation_High_Risk_Cases = np.std(data.loc[:,'High_Risk_Cases'])
min_High_Risk_Cases = np.min(data.loc[:,'High_Risk_Cases'])
max_High_Risk_Cases = np.max(data.loc[:,'High_Risk_Cases'])
print('Sum =',sum_High_Risk_Cases)
print('Range :- [',min_High_Risk_Cases,',',max_High_Risk_Cases,']')
print('Min =',min_High_Risk_Cases)
print('Max =',max_High_Risk_Cases)
print('Mean =',mean_High_Risk_Cases)
print('Median =',median_High_Risk_Cases)
print('Variance =',variance_High_Risk_Cases)
print('Standard Deviation =',standard_deviation_High_Risk_Cases,'\n\n')



print('Statistical analysis of column Compliance_Violations :-\n')
sum_Compliance_Violations = np.sum(data.loc[:,'Compliance_Violations'])
mean_Compliance_Violations = np.mean(data.loc[:,'Compliance_Violations'])
median_Compliance_Violations = np.median(data.loc[:,'Compliance_Violations'])
variance_Compliance_Violations = np.var(data.loc[:,'Compliance_Violations'])
standard_deviation_Compliance_Violations = np.std(data.loc[:,'Compliance_Violations'])
min_Compliance_Violations = np.min(data.loc[:,'Compliance_Violations'])
max_Compliance_Violations = np.max(data.loc[:,'Compliance_Violations'])
print('Sum =',sum_Compliance_Violations)
print('Range :- [',min_Compliance_Violations,',',max_Compliance_Violations,']')
print('Min =',min_Compliance_Violations)
print('Max =',max_Compliance_Violations)
print('Mean =',mean_Compliance_Violations)
print('Median =',median_Compliance_Violations)
print('Variance =',variance_Compliance_Violations)
print('Standard Deviation =',standard_deviation_Compliance_Violations,'\n\n')
```

```
Statistical analysis of column Total_Audit_Engagements :-

Sum = 278452
Range :- [ 603 , 4946 ]
Min = 603
Max = 4946
Mean = 2784.52
Median = 2650.0
Variance = 1626741.2095999997
Standard Deviation = 1275.4376541407266


Statistical analysis of column High_Risk_Cases :-

Sum = 27773
Range :- [ 51 , 500 ]
Min = 51
Max = 500
Mean = 277.73
Median = 293.0
Variance = 18239.797099999996
Standard Deviation = 135.05479295456342


Statistical analysis of column Compliance_Violations :-

Sum = 10548
Range :- [ 10 , 200 ]
Min = 10
Max = 200
Mean = 105.48
Median = 114.5
Variance = 3035.1896000000015
Standard Deviation = 55.09255485090523
```

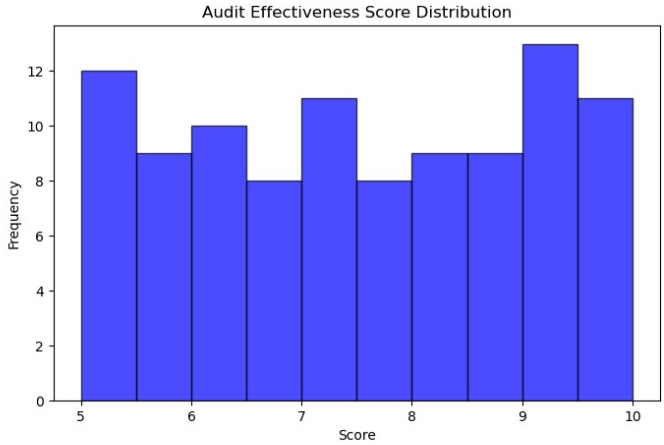
In [37]:
```
d4=data.copy()
print('Unique Years (columnn-1) :-\n',np.unique(d4.loc[:,'Year']))
print('Unique Firm_Name (columnn-2) :-\n',np.unique(d4.loc[:,'Firm_Name']))
print('Unique Total_Audit_Engagements (columnn-3) :-\n',np.unique(d4.loc[:,'Total_Audit_Engagements']))
print('Unique High_Risk_Cases (columnn-4) :-\n',np.unique(d4.loc[:,'High_Risk_Cases']))
print('Unique Compliance_Violations (columnn-5) :-\n',np.unique(d4.loc[:,'Compliance_Violations']))
print('Unique Fraud_Cases_Detected (columnn-6) :-\n',np.unique(d4.loc[:,'Fraud_Cases_Detected']))
print('Unique Industry_Affected (columnn-7) :-\n',np.unique(d4.loc[:,'Industry_Affected']))
print('Unique Total_Revenue_Impact (columnn-8) :-\n',np.unique(d4.loc[:,'Total_Revenue_Impact']))
print('Unique AI_Used_for_Auditing (columnn-9) :-\n',np.unique(d4.loc[:,'AI_Used_for_Auditing']))
print('Unique Employee_Workload (columnn-10) :-\n',np.unique(d4.loc[:,'Employee_Workload']))
print('Unique Audit_Effectiveness_Score (columnn-11) :-\n',np.unique(d4.loc[:,'Audit_Effectiveness_Score']))
print('Unique Client_Satisfaction_Score (columnn-12) :-\n',np.unique(d4.loc[:,'Client_Satisfaction_Score']))
```

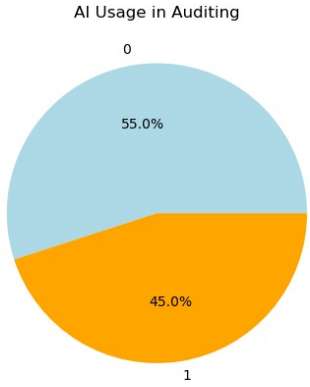
```
Unique Years (columnn-1) :-
 [2020 2021 2022 2023 2024 2025]
Unique Firm_Name (columnn-2) :-
 ['Deloitte' 'Ernst & Young' 'KPMG' 'PwC']
Unique Total_Audit_Engagements (columnn-3) :-
 [ 603  718  760  797  811  818  912  962  995 1012 1069 1076 1078 1119
 1199 1275 1286 1497 1516 1574 1578 1581 1695 1727 1760 1771 1810 1825
 1896 1924 1925 1992 2030 2119 2133 2183 2208 2238 2239 2245 2267 2283
 2333 2438 2490 2503 2506 2515 2556 2646 2654 2676 2680 2712 2804 2829
 2861 2885 2919 3076 3101 3230 3264 3305 3449 3490 3560 3571 3589 3616
 3630 3810 3852 3958 3981 4092 4103 4156 4179 4287 4324 4327 4340 4390
 4401 4452 4470 4473 4481 4570 4595 4601 4606 4624 4773 4775 4784 4813
 4895 4946]
Unique High_Risk_Cases (columnn-4) :-
 [ 51  52  58  59  62  63  65  74  77  78  86  91 101 102 110 112 122 123
 127 133 148 158 164 176 185 189 195 199 200 201 208 212 216 221 222 226
 231 242 247 249 258 273 284 285 290 296 302 306 312 313 317 325 330 345
 354 356 358 360 362 367 377 380 382 391 395 397 398 406 408 415 423 425
 428 433 435 442 448 449 453 463 469 472 481 483 487 497 500]
Unique Compliance_Violations (columnn-5) :-
 [ 10  15  17  19  20  25  30  31  35  36  37  38  39  42  45  46  47  48
  49  53  55  58  61  65  67  71  73  74  81  82  86  87  90  92  94  99
 100 102 103 114 115 116 121 123 124 125 126 132 134 136 137 138 139 141
 142 144 146 149 151 152 153 155 158 166 167 169 173 179 181 183 184 185
 186 193 196 198 199 200]
Unique Fraud_Cases_Detected (columnn-6) :-
 [  5   6  10  11  12  13  14  17  18  20  21  22  24  25  26  27  28  30
  32  33  35  36  37  39  40  42  45  46  48  50  51  52  54  56  58  59
  60  61  62  63  64  66  67  69  70  71  72  73  79  80  81  82  86  87
  90  91  92  93  94  95  96  97  99 100]
Unique Industry_Affected (columnn-7) :-
 ['Finance' 'Healthcare' 'Retail' 'Tech']
Unique Total_Revenue_Impact (columnn-8) :-
 [ 33.46  48.    52.64  53.85  54.07  61.17  65.    69.97  83.61  85.46
  88.08  89.79  94.56  95.68 104.98 106.93 110.06 114.24 118.92 129.98
 130.85 131.83 139.48 140.29 150.59 156.76 156.98 160.31 168.15 172.
 178.86 182.06 182.9  193.07 197.87 206.47 213.92 216.84 221.42 224.92
 225.42 228.15 229.11 235.12 240.87 243.85 249.68 250.74 258.49 263.14
 265.76 268.67 276.3  284.84 285.51 291.   291.77 294.38 302.28 307.61
 307.88 318.79 320.75 334.56 339.08 349.04 362.31 378.3  381.61 382.67
 388.5  389.31 389.37 395.59 403.02 415.3  418.49 424.03 426.07 429.95
 432.8  435.76 438.45 438.89 440.9  444.51 445.62 447.14 454.65 456.08
 461.33 468.13 468.82 474.21 474.32 478.   483.07 485.64 495.19 497.06]
Unique AI_Used_for_Auditing (columnn-9) :-
 ['No' 'Yes']
Unique Employee_Workload (columnn-10) :-
 [40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
 64 65 66 67 68 69 70 71 73 74 75 76 77 78 79 80]
Unique Audit_Effectiveness_Score (columnn-11) :-
 [ 5.   5.1  5.2  5.3  5.4  5.5  5.6  5.8  5.9  6.   6.1  6.2  6.3  6.4
  6.5  6.6  6.7  6.8  6.9  7.   7.2  7.3  7.4  7.5  7.6  7.7  7.8  7.9
  8.   8.2  8.3  8.4  8.5  8.6  8.7  8.8  8.9  9.   9.1  9.2  9.3  9.4
  9.5  9.6  9.8  9.9 10. ]
Unique Client_Satisfaction_Score (columnn-12) :-
 [ 5.   5.1  5.2  5.3  5.4  5.5  5.6  5.7  5.8  5.9  6.   6.1  6.2  6.3
  6.4  6.5  6.6  6.7  6.9  7.   7.1  7.3  7.4  7.5  7.6  7.7  7.8  7.9
  8.   8.3  8.4  8.5  8.6  8.7  8.8  8.9  9.   9.1  9.2  9.3  9.5  9.6
  9.9 10. ]
```

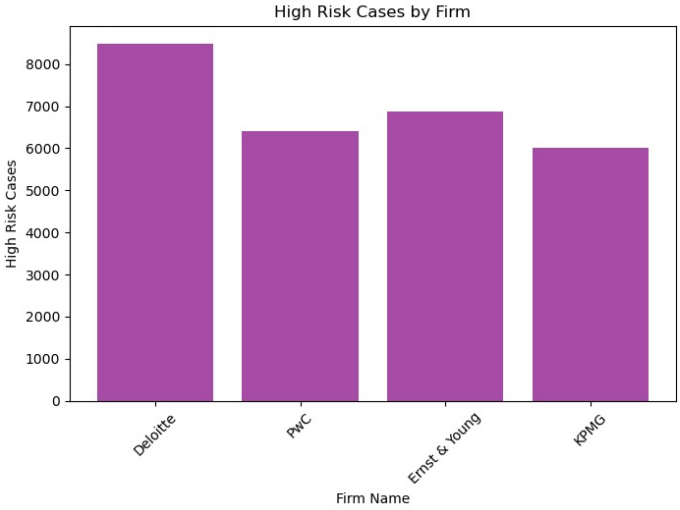
In [45]:

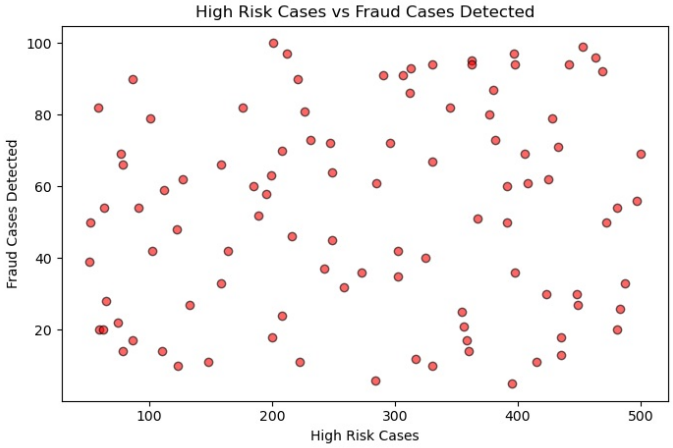
```python
plt.figure(figsize=(14, 10))

# 1. Histogram - Audit Effectiveness Score
plt.subplot(2, 2, 1)
plt.hist(df["Audit_Effectiveness_Score"], bins=10, color='blue', edgecolor='black', alpha=0.7)
plt.title("Audit Effectiveness Score Distribution")
plt.xlabel("Score")
plt.ylabel("Frequency")

# 2. Pie Chart - AI Usage Distribution
plt.subplot(2, 2, 2)
ai_counts = df["AI_Used_for_Auditing"].value_counts()
plt.pie(ai_counts, labels=ai_counts.index, autopct='%1.1f%%', colors=["lightblue", "orange"])
plt.title("AI Usage in Auditing")

# 3. Bar Chart - High Risk Cases by Firm
plt.subplot(2, 2, 3)
firms = df["Firm_Name"].value_counts().index
high_risk_cases = df.groupby("Firm_Name")["High_Risk_Cases"].sum()
plt.bar(firms, high_risk_cases, color='purple', alpha=0.7)
plt.title("High Risk Cases by Firm")
plt.xlabel("Firm Name")
plt.ylabel("High Risk Cases")
plt.xticks(rotation=45)
```

```
# 4. Scatter Plot - Fraud Cases Detected vs. High Risk Cases
plt.subplot(2, 2, 4)
plt.scatter(df["High_Risk_Cases"], df["Fraud_Cases_Detected"], c="red", alpha=0.6, edgecolors="black")
plt.title("High Risk Cases vs Fraud Cases Detected")
plt.xlabel("High Risk Cases")
plt.ylabel("Fraud Cases Detected")

# Show plots
plt.tight_layout()
plt.show()
```









In [89]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

label_encoder = LabelEncoder()
df["AI_Used_for_Auditing"] = label_encoder.fit_transform(df["AI_Used_for_Auditing"])

X = df.select_dtypes(include=[np.number]).drop(columns=["AI_Used_for_Auditing"])
y = df["AI_Used_for_Auditing"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.314, random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train_scaled, y_train)

y_pred = knn.predict(X_test_scaled)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred, average='weighted'))
print("Recall:", recall_score(y_test, y_pred, average='weighted'))
print("F1 Score:", f1_score(y_test, y_pred, average='weighted'))
```

```
Accuracy: 0.5
Precision: 0.5235294117647059
Recall: 0.5
F1 Score: 0.503921568627451
```

In [ ]:

In [ ]: