Name: Mihir Unmesh Patil

Roll No: TYCOC213

Batch: C/ C-3

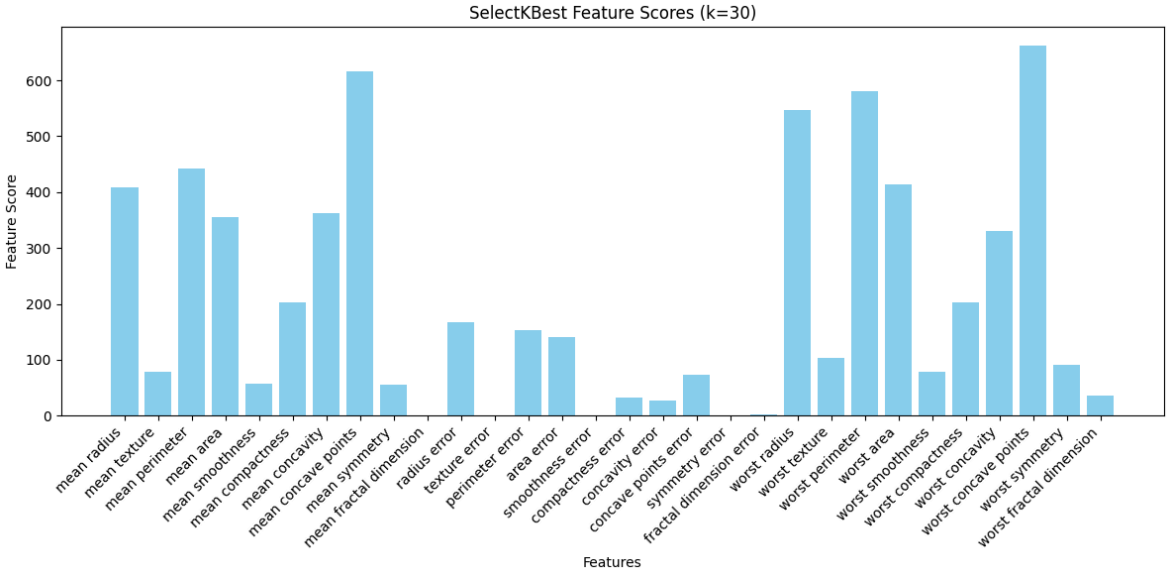
In [3]:
```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectKBest, SelectPercentile, f_classif
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

data = load_breast_cancer()

X = data.data
y = data.target
features = data.feature_names

print(pd.DataFrame(data.data, columns=data.feature_names).head())

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_

print("\nThe feature names are: \n")
for i in features:
    print(i)
print()
```

```
   mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
0        17.99         10.38          122.80     1001.0          0.11840
1        20.57         17.77          132.90     1326.0          0.08474
2        19.69         21.25          130.00     1203.0          0.10960
3        11.42         20.38           77.58      386.1          0.14250
4        20.29         14.34          135.10     1297.0          0.10030

   mean compactness  mean concavity  mean concave points  mean symmetry  \
0           0.27760          0.3001              0.14710         0.2419
1           0.07864          0.0869              0.07017         0.1812
2           0.15990          0.1974              0.12790         0.2069
3           0.28390          0.2414              0.10520         0.2597
4           0.13280          0.1980              0.10430         0.1809

   mean fractal dimension  ...  worst radius  worst texture  worst perimeter  \
0                 0.07871  ...         25.38          17.33           184.60
1                 0.05667  ...         24.99          23.41           158.80
2                 0.05999  ...         23.57          25.53           152.50
3                 0.09744  ...         14.91          26.50            98.87
4                 0.05883  ...         22.54          16.67           152.20

   worst area  worst smoothness  worst compactness  worst concavity  \
0      2019.0            0.1622             0.6656           0.7119
1      1956.0            0.1238             0.1866           0.2416
2      1709.0            0.1444             0.4245           0.4504
3       567.7            0.2098             0.8663           0.6869
4      1575.0            0.1374             0.2050           0.4000

   worst concave points  worst symmetry  worst fractal dimension
0                0.2654          0.4601                  0.11890
1                0.1860          0.2750                  0.08902
2                0.2430          0.3613                  0.08758
3                0.2575          0.6638                  0.17300
4                0.1625          0.2364                  0.07678

[5 rows x 30 columns]

The feature names are:

mean radius
mean texture
mean perimeter
mean area
mean smoothness
mean compactness
mean concavity
mean concave points
mean symmetry
mean fractal dimension
radius error
texture error
perimeter error
area error
smoothness error
compactness error
concavity error
concave points error
symmetry error
fractal dimension error
worst radius
```

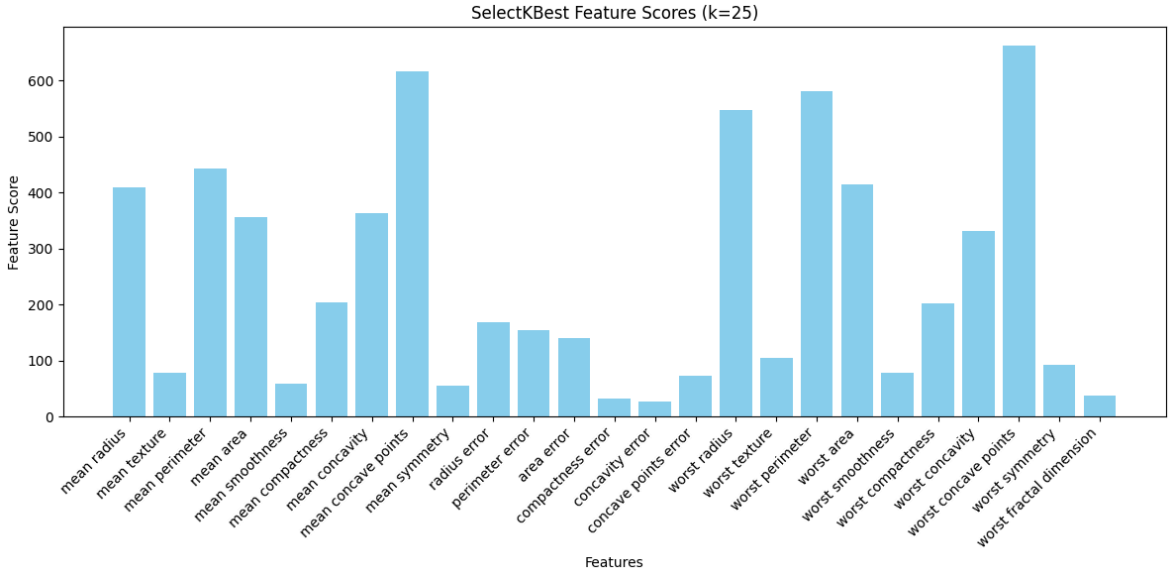
```
worst texture
worst perimeter
worst area
worst smoothness
worst compactness
worst concavity
worst concave points
worst symmetry
worst fractal dimension
```

In [4]:
```python
k_values = [30, 25, 10]
kbest_features_all = []

for k in k_values:
    select_k_best = SelectKBest(f_classif, k=k)
    X_train_kbest = select_k_best.fit_transform(X_train, y_train)
    kbest_indices = select_k_best.get_support(indices=True)
    kbest_features = features[kbest_indices]
    kbest_features_all.append(kbest_features)

    print(f"Selected features using SelectKBest (k={k}):", kbest_indices)

    feature_scores = select_k_best.scores_
    kbest_scores = feature_scores[kbest_indices]

    plt.figure(figsize=(12, 6))
    plt.bar(kbest_features, kbest_scores, color='skyblue')
    plt.xlabel("Features")
    plt.ylabel("Feature Score")
    plt.title(f"SelectKBest Feature Scores (k={k})")
    plt.xticks(rotation=45, ha="right")
    plt.tight_layout()
    plt.show()
```

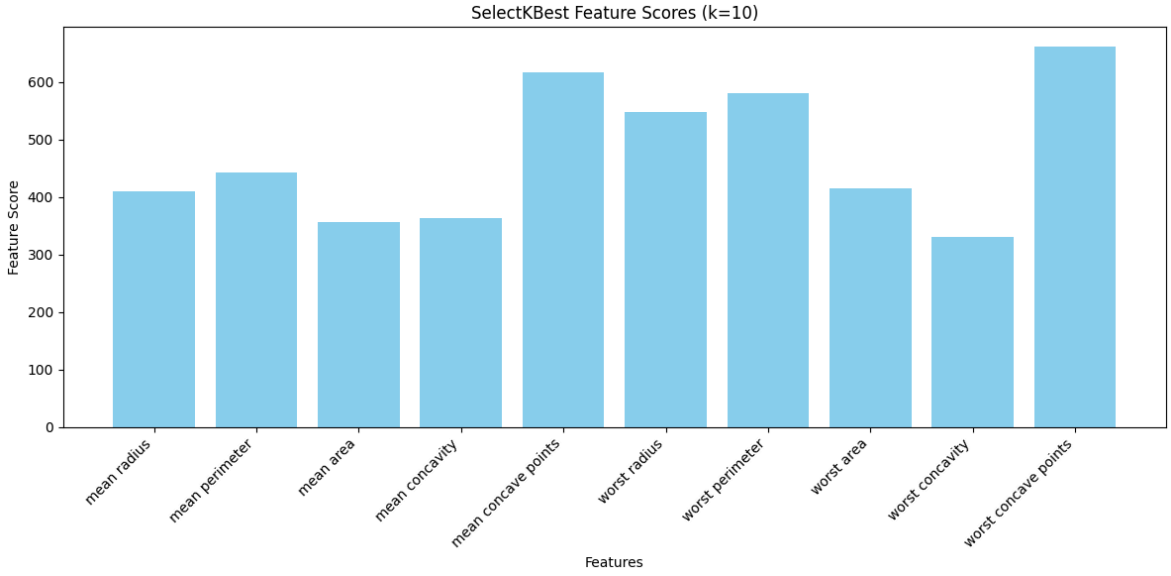
```
Selected features using SelectKBest (k=30): [ 0  1  2  3  4  5  6  7  8  9 10 11
12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29]
```




```
Selected features using SelectKBest (k=25): [ 0  1  2  3  4  5  6  7  8 10 12 13
15 16 17 20 21 22 23 24 25 26 27 28
 29]
```

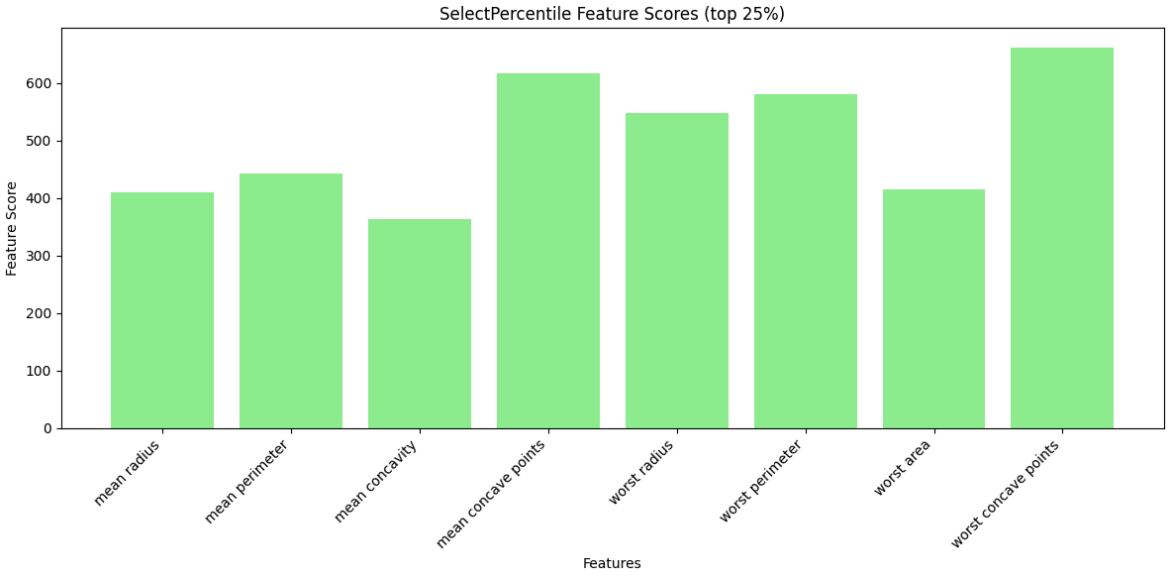



```
Selected features using SelectKBest (k=10): [ 0  2  3  6  7 20 22 23 26 27]
```

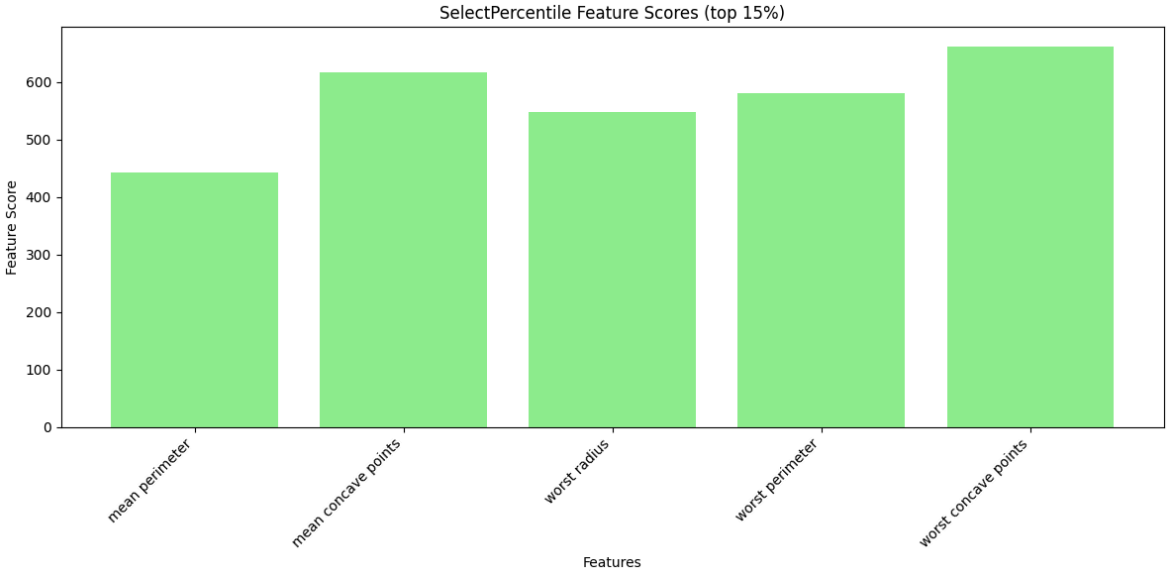


In [5]:
```python
percentiles = [25, 15, 5]
percentile_features_all = []

for percentile in percentiles:
    select_percentile = SelectPercentile(f_classif, percentile=percentile)
    X_train_percentile = select_percentile.fit_transform(X_train, y_train)
    percentile_indices = select_percentile.get_support(indices=True)
    percentile_features = features[percentile_indices]
    percentile_features_all.append(percentile_features)

    print(f"\nSelected features using SelectPercentile (top {percentile}%):", pe

    feature_scores = select_percentile.scores_
    percentile_scores = feature_scores[percentile_indices]

    plt.figure(figsize=(12, 6))
    plt.bar(percentile_features, percentile_scores, color='lightgreen')
    plt.xlabel("Features")
    plt.ylabel("Feature Score")
    plt.title(f"SelectPercentile Feature Scores (top {percentile}%)")
    plt.xticks(rotation=45, ha="right")
    plt.tight_layout()
    plt.show()
```

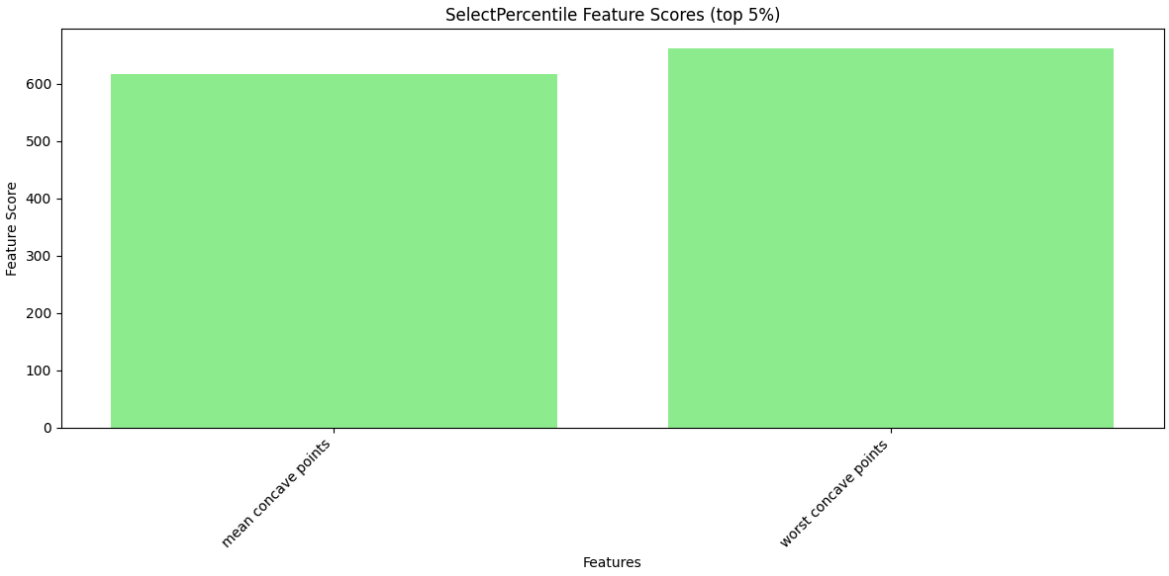
```
Selected features using SelectPercentile (top 25%): [ 0  2  6  7 20 22 23 27]
```




```
Selected features using SelectPercentile (top 15%): [ 2  7 20 22 27]
```



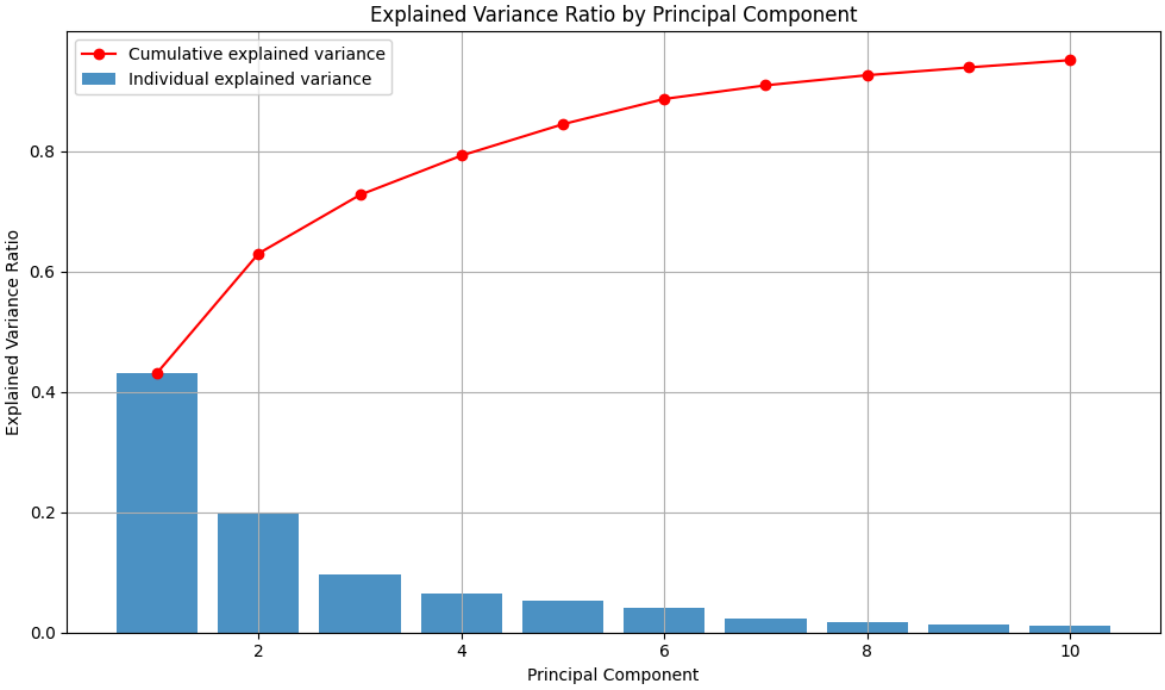

```
Selected features using SelectPercentile (top 5%): [ 7 27]
```



In [6]:
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)

pca = PCA(n_components=10)
```

```
X_train_pca = pca.fit_transform(X_train_scaled)

print("\nExplained variance ratio by each principal component:")
explained_variance_ratio = pca.explained_variance_ratio_
print(explained_variance_ratio)

cumulative_variance = np.cumsum(explained_variance_ratio)

plt.figure(figsize=(10, 6))
plt.bar(range(1, len(explained_variance_ratio) + 1), explained_variance_ratio, a
plt.plot(range(1, len(explained_variance_ratio) + 1), cumulative_variance, c='re
plt.title('Explained Variance Ratio by Principal Component')
plt.xlabel('Principal Component')
plt.ylabel('Explained Variance Ratio')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

loadings = pca.components_.T * np.sqrt(pca.explained_variance_)

feature_names = data.feature_names
num_pc = pca.n_components_

pc_list = [f"PC{i+1}" for i in range(num_pc)]
loadings_df = pd.DataFrame.from_records(loadings, columns=pc_list, index=feature
print("\nFeature Loadings DataFrame:")
print(loadings_df)
```

```
Explained variance ratio by each principal component:
[0.4316748  0.19845652 0.09733159 0.06531574 0.05212151 0.0419896
 0.02263461 0.01682669 0.0129469  0.0120941 ]
```



Feature Loadings DataFrame:

	PC1	PC2	PC3	PC4	PC5
mean radius	0.781626	-0.580723	-0.016703	-0.057967	-0.058744
mean texture	0.384967	-0.108653	0.109308	0.848158	0.004710
mean perimeter	0.814411	-0.537733	-0.017759	-0.060669	-0.058994
mean area	0.791236	-0.571358	0.044957	-0.082122	-0.015631
mean smoothness	0.511556	0.439480	-0.216256	-0.181252	0.494682
mean compactness	0.860189	0.388050	-0.124556	-0.058776	-0.016800
mean concavity	0.938113	0.145676	0.007813	-0.055790	-0.114766
mean concave points	0.949970	-0.092773	-0.065102	-0.088716	0.058869
mean symmetry	0.533505	0.439069	-0.080282	-0.024195	0.334321
mean fractal dimension	0.211486	0.881212	-0.034640	-0.104849	0.079687
radius error	0.739204	-0.282704	0.442281	-0.127302	0.230753
texture error	0.131407	0.242938	0.612030	0.516247	0.242359
perimeter error	0.754699	-0.233555	0.449617	-0.114842	0.176918
area error	0.718021	-0.383402	0.363575	-0.152418	0.202506
smoothness error	0.093451	0.482449	0.503728	-0.063232	0.307162
compactness error	0.598066	0.573337	0.285518	0.023481	-0.332657
concavity error	0.562154	0.482251	0.354098	-0.038360	-0.435233
concave points error	0.659433	0.336621	0.395899	-0.080338	-0.227869
symmetry error	0.194704	0.384686	0.443444	-0.030343	0.343780
fractal dimension error	0.371871	0.691075	0.378946	-0.080640	-0.302946
worst radius	0.815428	-0.545655	-0.093107	-0.017645	-0.012451
worst texture	0.386913	-0.079636	-0.085563	0.897897	0.047710
worst perimeter	0.850515	-0.496398	-0.093721	-0.014569	-0.031827
worst area	0.806904	-0.542011	-0.033380	-0.040741	0.021110
worst smoothness	0.451683	0.415669	-0.480257	-0.010265	0.405064
worst compactness	0.746080	0.374791	-0.394643	0.107719	-0.172755
worst concavity	0.839498	0.241504	-0.290567	0.058133	-0.251341
worst concave points	0.909054	-0.006249	-0.311620	0.005224	-0.058248
worst symmetry	0.441272	0.297348	-0.489124	0.093870	0.253374
worst fractal dimension	0.450596	0.695320	-0.378545	0.031910	-0.106988

	PC6	PC7	PC8	PC9	PC10
mean radius	0.035375	-0.125581	0.028554	0.117446	0.034773
mean texture	-0.082695	0.022130	0.054137	-0.162481	0.000343
mean perimeter	0.030079	-0.116905	0.022759	0.115283	0.021919
mean area	0.001484	-0.067536	0.058082	0.113781	0.060858
mean smoothness	-0.296659	-0.103669	-0.200117	0.032833	-0.129803
mean compactness	-0.029967	0.013052	-0.059294	0.060008	-0.127595
mean concavity	-0.031522	-0.074085	-0.058343	0.017482	-0.037974
mean concave points	-0.048317	-0.149589	-0.075451	0.051492	-0.056760
mean symmetry	0.400659	-0.085091	-0.259088	-0.128150	0.322722
mean fractal dimension	-0.151048	0.210083	-0.075507	0.120080	0.027318
radius error	-0.043620	0.274977	-0.026370	-0.118956	-0.004997
texture error	-0.011770	-0.067651	-0.220459	0.325137	-0.027010
perimeter error	-0.018254	0.298818	-0.035644	-0.123775	-0.076829
area error	-0.074303	0.302803	0.033785	-0.104756	0.020795
smoothness error	-0.339452	-0.284188	0.402674	-0.047649	0.142292
compactness error	0.085849	0.017511	0.143812	-0.001056	-0.058530
concavity error	0.028669	-0.109043	-0.064830	-0.119896	0.058792
concave points error	-0.010983	-0.255684	-0.163482	-0.219078	-0.096097
symmetry error	0.589779	-0.062166	0.221000	0.094835	-0.232717
fractal dimension error	-0.043002	0.089347	0.027291	0.136707	0.236988
worst radius	0.004988	-0.031522	0.040568	0.067712	0.073181
worst texture	-0.086536	0.037718	0.002091	-0.035985	0.022257
worst perimeter	0.010803	-0.018760	0.037113	0.059374	0.044825
worst area	-0.036401	0.025655	0.065708	0.064995	0.111644
worst smoothness	-0.381010	-0.087295	0.116301	-0.079164	0.016122
worst compactness	0.051030	0.135593	0.118537	0.014110	-0.132609

```
worst concavity          0.007204  0.001603  0.029703 -0.036797 -0.068356
worst concave points    -0.016461 -0.119173 -0.032408 -0.066288 -0.073742
worst symmetry           0.576830  0.008071  0.118698 -0.062711  0.084095
worst fractal dimension -0.085908  0.284688  0.083370  0.127352  0.080800
```

In [7]:
```python
feature_counts = {}

for kbest_features in kbest_features_all:
    for feature in kbest_features:
        feature_counts[feature] = feature_counts.get(feature, 0) + 1

for percentile_features in percentile_features_all:
    for feature in percentile_features:
        feature_counts[feature] = feature_counts.get(feature, 0) + 1

feature_names = list(feature_counts.keys())
counts = list(feature_counts.values())

plt.figure(figsize=(14, 7))
plt.bar(feature_names, counts, color=['skyblue', 'lightgreen'])
plt.xlabel("Features")
plt.ylabel("Selection Count")
plt.title("Feature Selection Comparison: SelectKBest vs SelectPercentile")
plt.xticks(rotation=45, ha="right")
plt.tight_layout()
plt.show()
```