EDA

October 31, 2022

[6]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

[7]:
```
import warnings
warnings.filterwarnings('ignore')
```

1 Loading in the Dataset (Preprocess)

[9]:
```
df = pd.read_csv('data/cleaned_merged_seasons.csv', index_col=0)
df = df.sort_values(['name', 'season_x', 'GW']).set_index(['total_points',␣
↪'name'], drop = True).groupby('name', as_index=False).shift().dropna(subset␣
↪= ['season_x']).reset_index()
```

2 Exploratory Data Analysis

Let's analyze the features to see which ones are categorical versus which ones are numerical

[83]:
```
# Find which features are numerical vs. categorical for EDA

num_vars = []
cat_vars = []
num_cat_vars = []

for column in list(df.columns):

    if(df[column].dtype == 'float64' and len(df[column].unique())>=10):
        num_vars.append(column)
    elif(df[column].dtype == 'float64'):
        num_cat_vars.append(column)
    else:
        cat_vars.append(column)

    print("{} has {} unique values of type {}".format(column, len(df[column].
↪unique()), df[column].dtype))
```

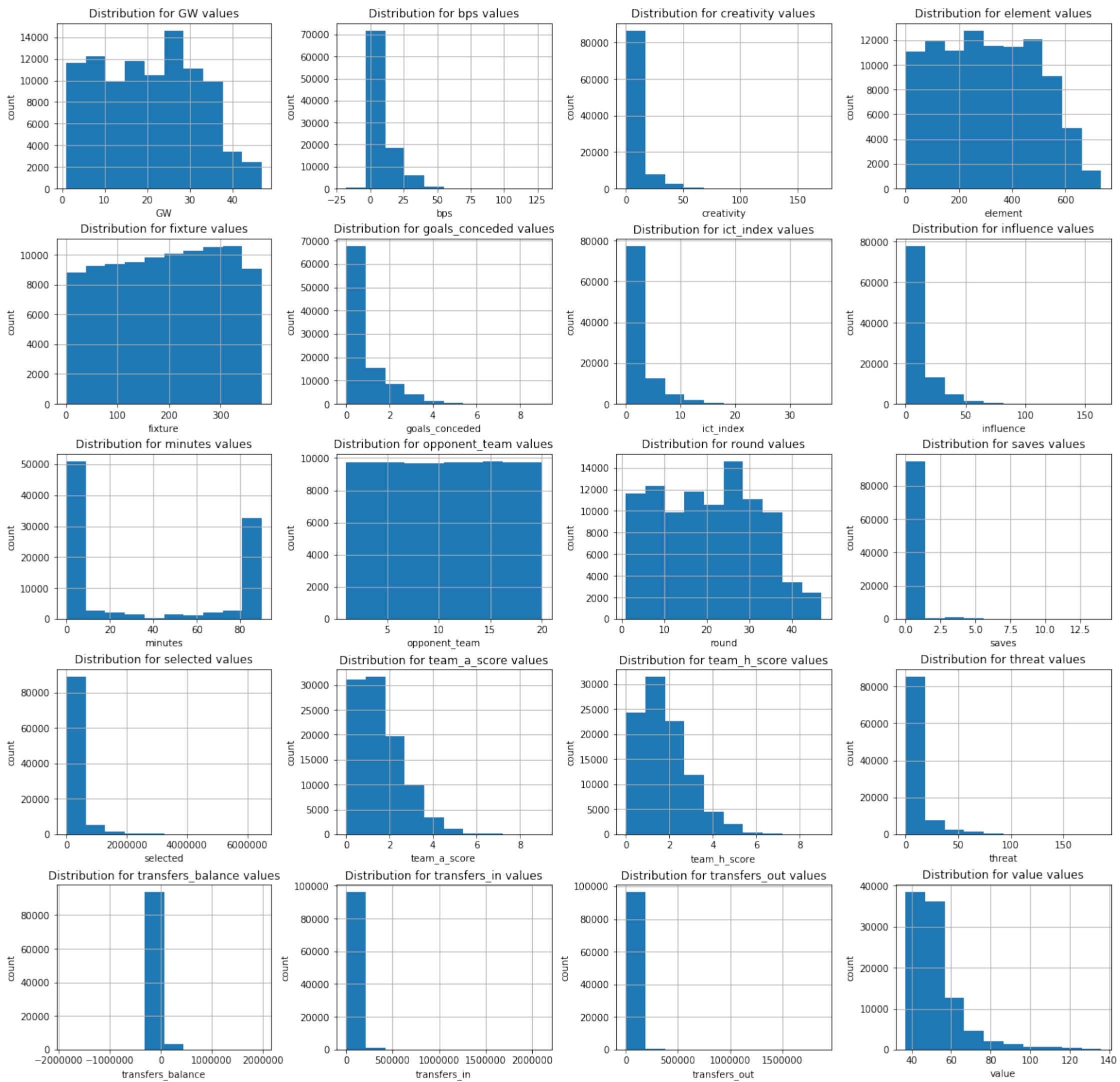
```
cat_vars.remove('total_points')
```

```
total_points has 31 unique values of type int64
name has 982 unique values of type object
season_x has 6 unique values of type object
position has 4 unique values of type object
team_x has 24 unique values of type object
assists has 5 unique values of type float64
bonus has 4 unique values of type float64
bps has 113 unique values of type float64
clean_sheets has 2 unique values of type float64
creativity has 860 unique values of type float64
element has 734 unique values of type float64
fixture has 380 unique values of type float64
goals_conceded has 10 unique values of type float64
goals_scored has 5 unique values of type float64
ict_index has 273 unique values of type float64
influence has 528 unique values of type float64
kickoff_time has 1428 unique values of type object
minutes has 91 unique values of type float64
opponent_team has 20 unique values of type float64
opp_team_name has 31 unique values of type object
own_goals has 2 unique values of type float64
penalties_missed has 2 unique values of type float64
penalties_saved has 3 unique values of type float64
red_cards has 2 unique values of type float64
round has 47 unique values of type float64
saves has 14 unique values of type float64
selected has 65272 unique values of type float64
team_a_score has 10 unique values of type float64
team_h_score has 11 unique values of type float64
threat has 149 unique values of type float64
transfers_balance has 32072 unique values of type float64
transfers_in has 24245 unique values of type float64
transfers_out has 26631 unique values of type float64
value has 100 unique values of type float64
was_home has 2 unique values of type object
yellow_cards has 2 unique values of type float64
GW has 47 unique values of type float64
```

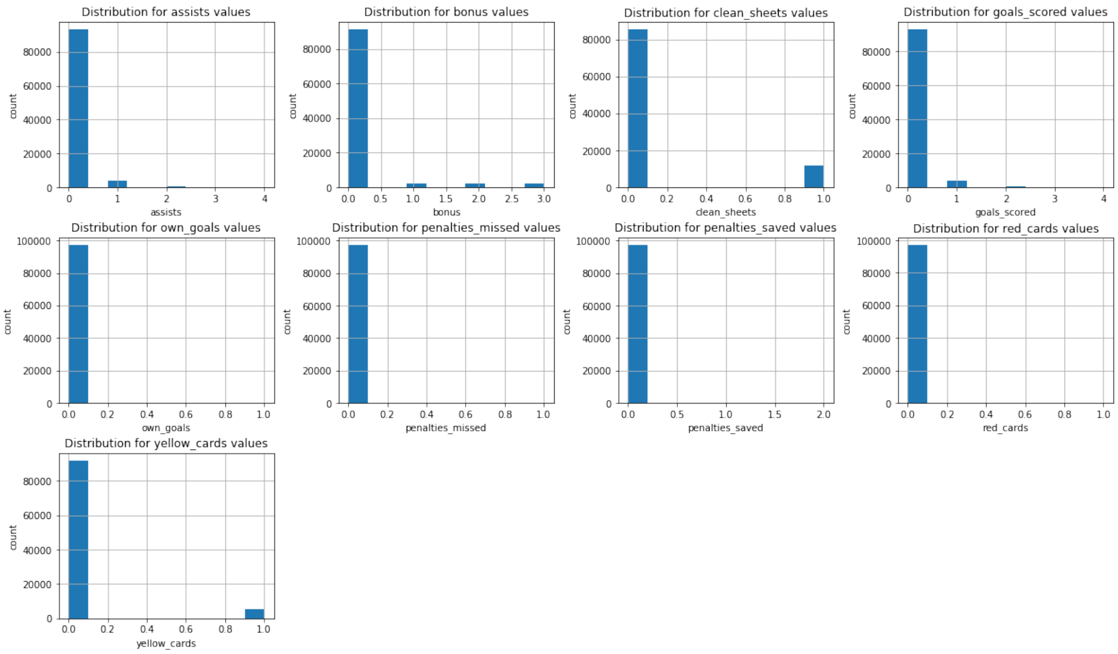
Examining the distribution for numerical features, numerical features with a small number of categories, and categorical features …

[74]:
```
# Examine the distribution for numerical features (floats with 10+ unique␣
↪values)

hist = df.hist(column=num_vars, layout=(5, 4), figsize=(20,20))
```

```
for ax, column_name in zip(hist.flatten(), sorted(num_vars)):
    ax.set_title("Distribution for {} values".format(column_name))
    ax.set_xlabel(column_name)
    ax.set_ylabel('count')
```



[90]:
```
# Examine the distribution for numerical categorical features (floats with <10␣
↪unique values)

hist = df.hist(column=num_cat_vars, layout=(5, 4), figsize=(20,20))
for ax, column_name in zip(hist.flatten(), sorted(num_cat_vars)):
    ax.set_title("Distribution for {} values".format(column_name))
    ax.set_xlabel(column_name)
    ax.set_ylabel('count')
```

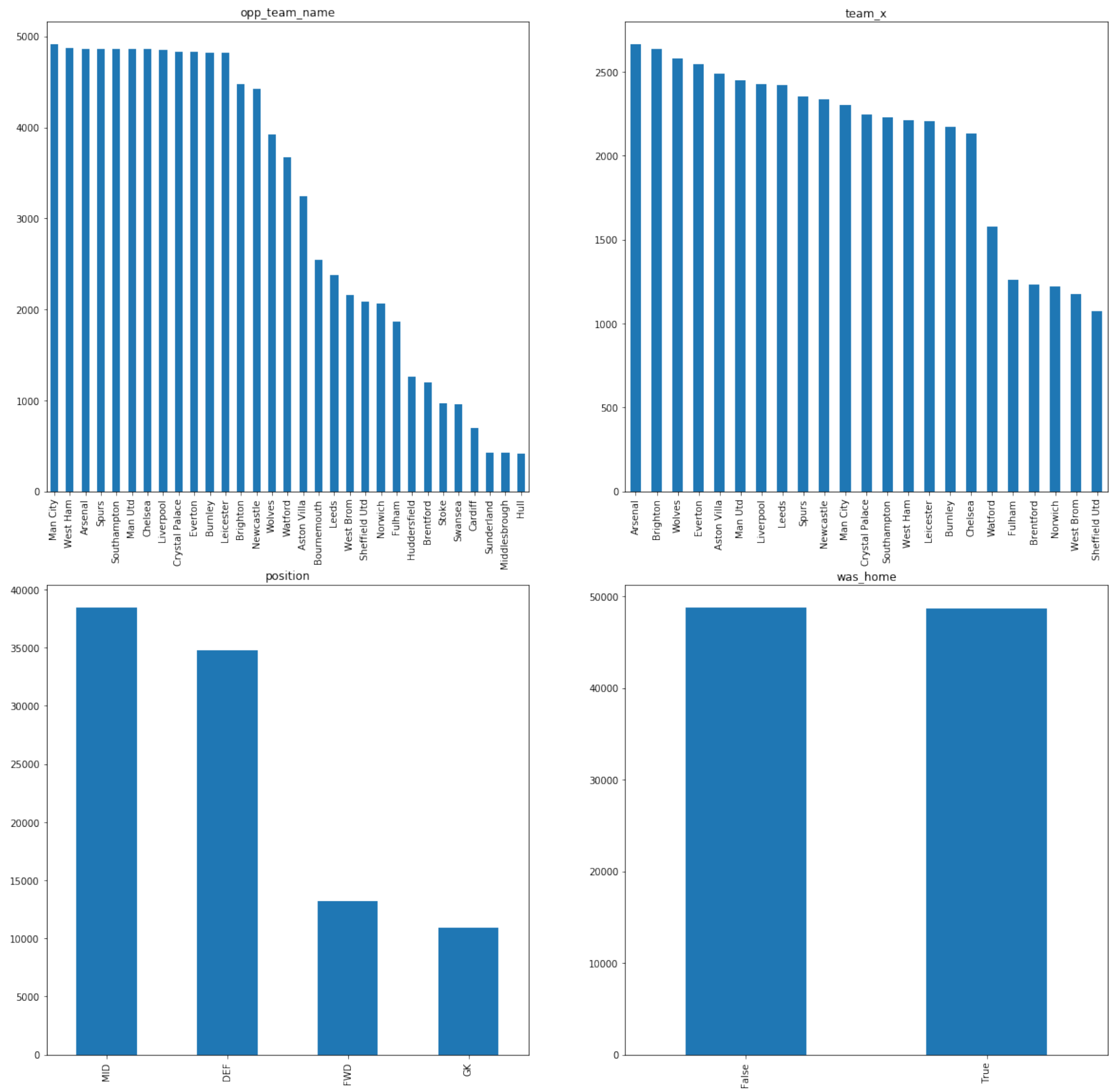


[ ]:

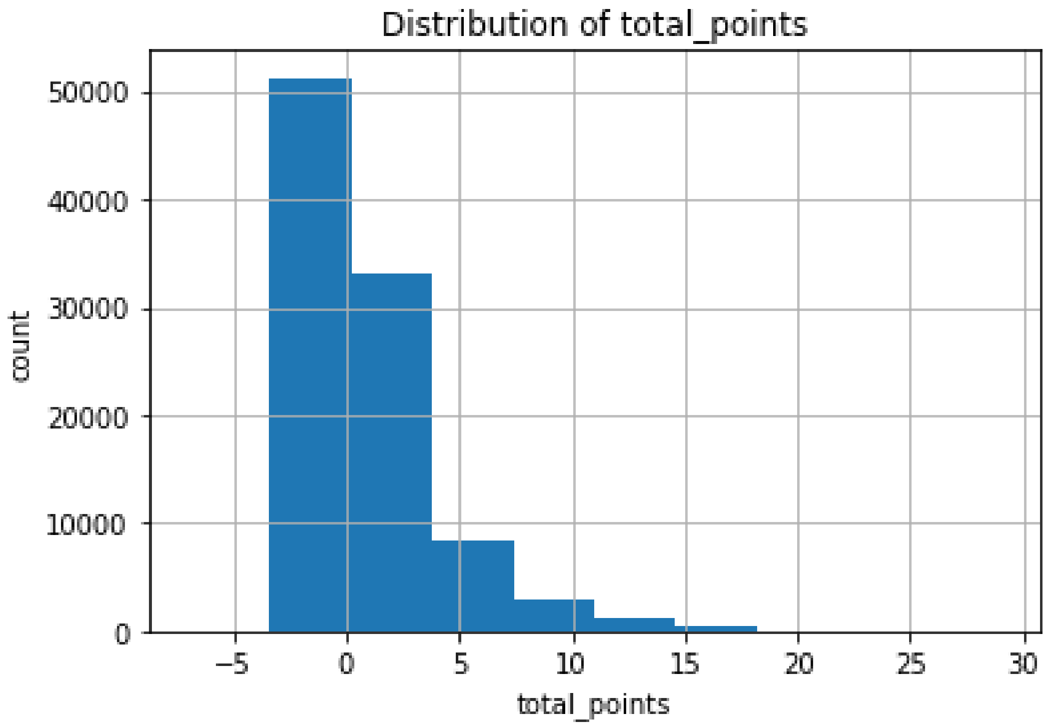
```
# Drop these as they are not categorical features we want to consider (just got
↪included beacuse they are not floats)

for feature in ['name', 'season_x', 'kickoff_time']:
    cat_vars.remove(feature)
```

[104]:

```
# Examine the distribution for categorical features (non float datatypes that
↪aren't any of the above)

fig, axes = plt.subplots(nrows=2,ncols=2, figsize=(20,20))

for index, feature in enumerate(sorted(cat_vars)):
    df[feature].value_counts().plot(ax=axes[index%2,index//2], subplots=True,
↪kind='bar')
```



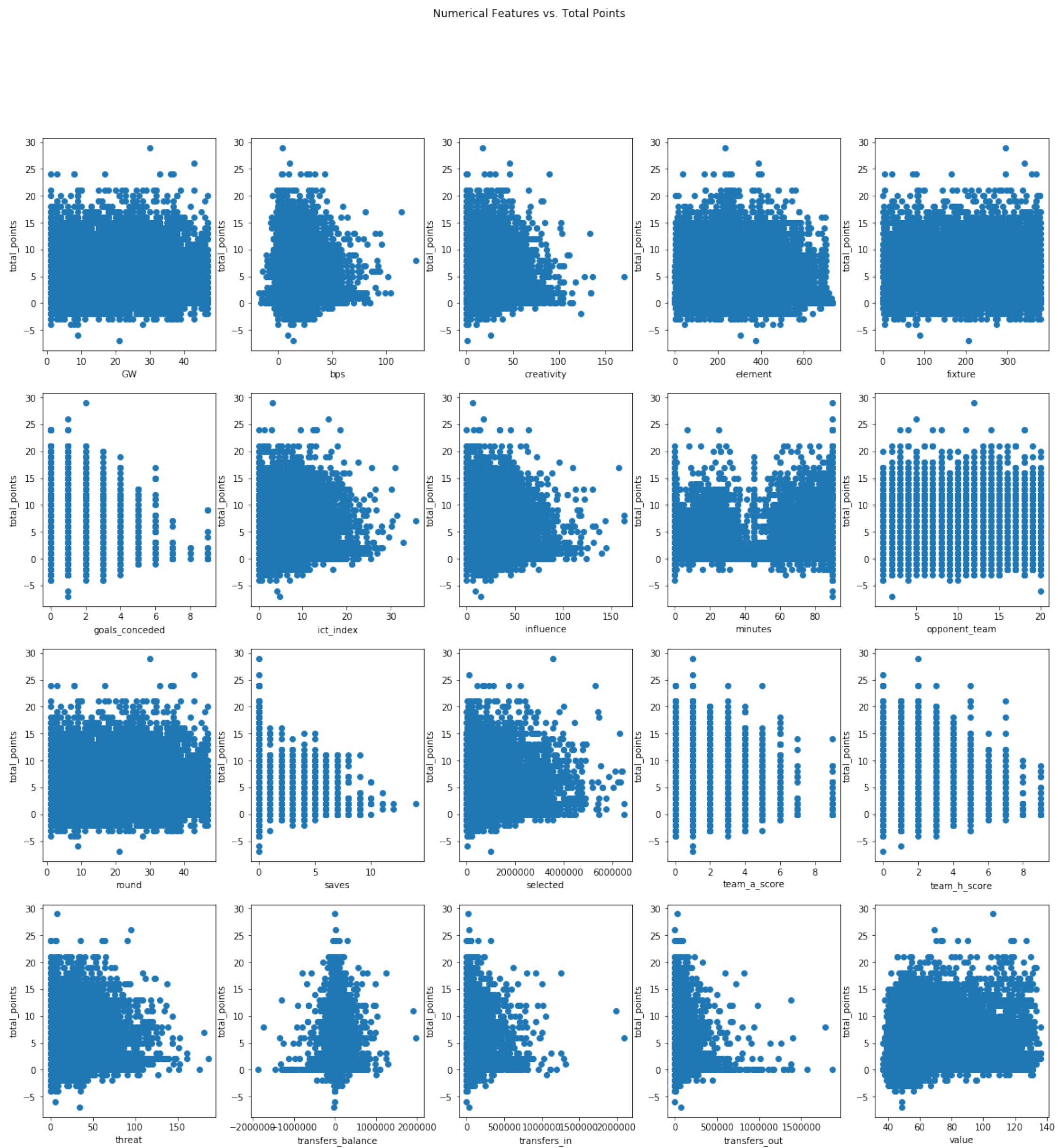
[25]:
```
# Examine the distribution of y-labels

df.hist(column='total_points')
plt.title('Distribution of total_points')
plt.xlabel('total_points')
plt.ylabel('count')
plt.show()
```



Let's look at the correlation between features and target …

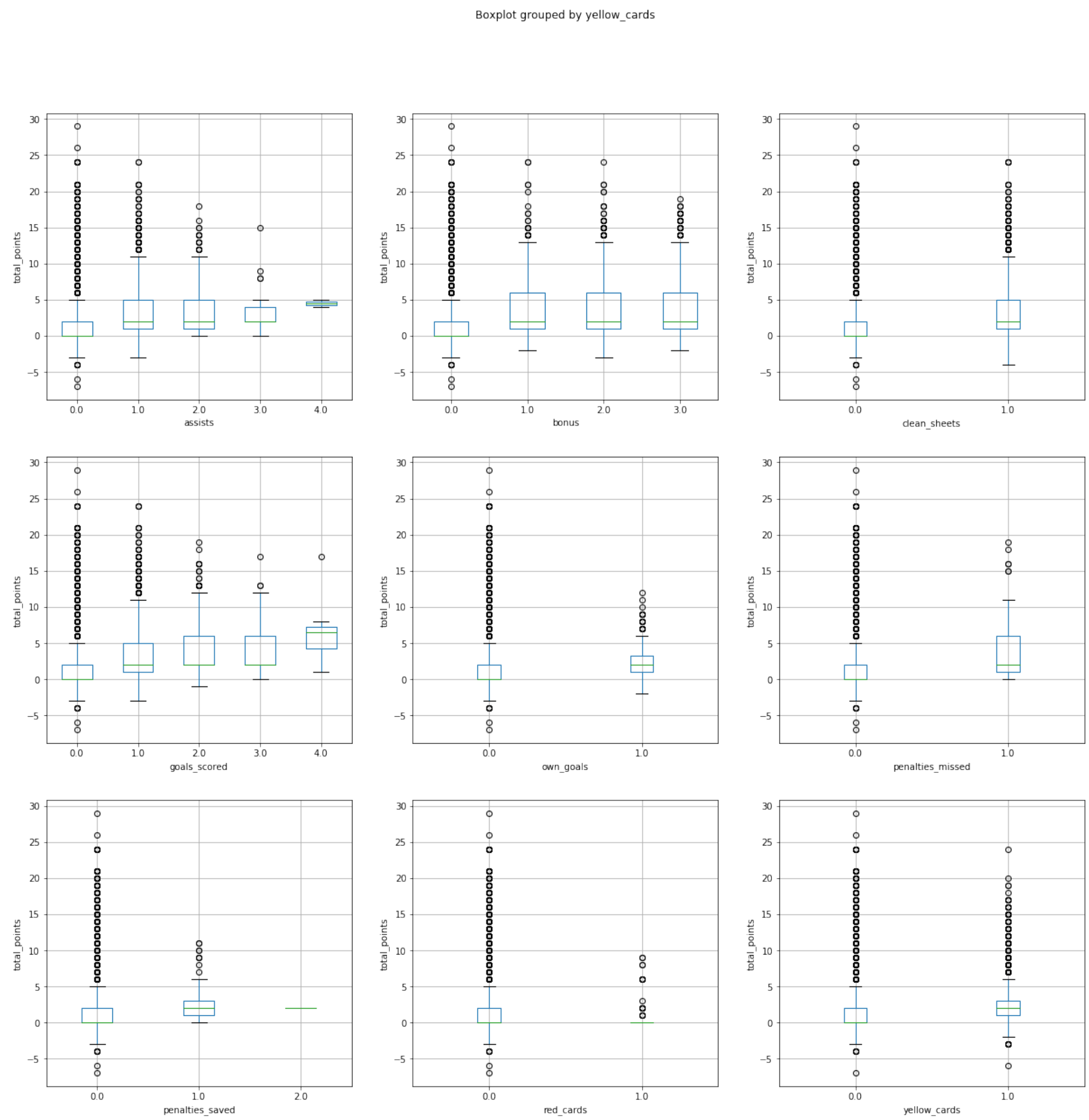
[106]:
```
df_x = df.drop(columns=['total_points'])
df_y = df['total_points']
```

[116]:
```
# Examine the correlation with the target for numerical features (floats with␣
↪10+ unique values)

fig, axes = plt.subplots(4, 5, figsize=(20,20))
fig.suptitle('Numerical Features vs. Total Points')

for index, feature in enumerate(sorted(num_vars)):
    x_subplot = index//5
    y_subplot = index%5
    axes[x_subplot, y_subplot].scatter(df_x[feature], df_y)
    axes[x_subplot, y_subplot].set_xlabel(feature)
    axes[x_subplot, y_subplot].set_ylabel('total_points')
```



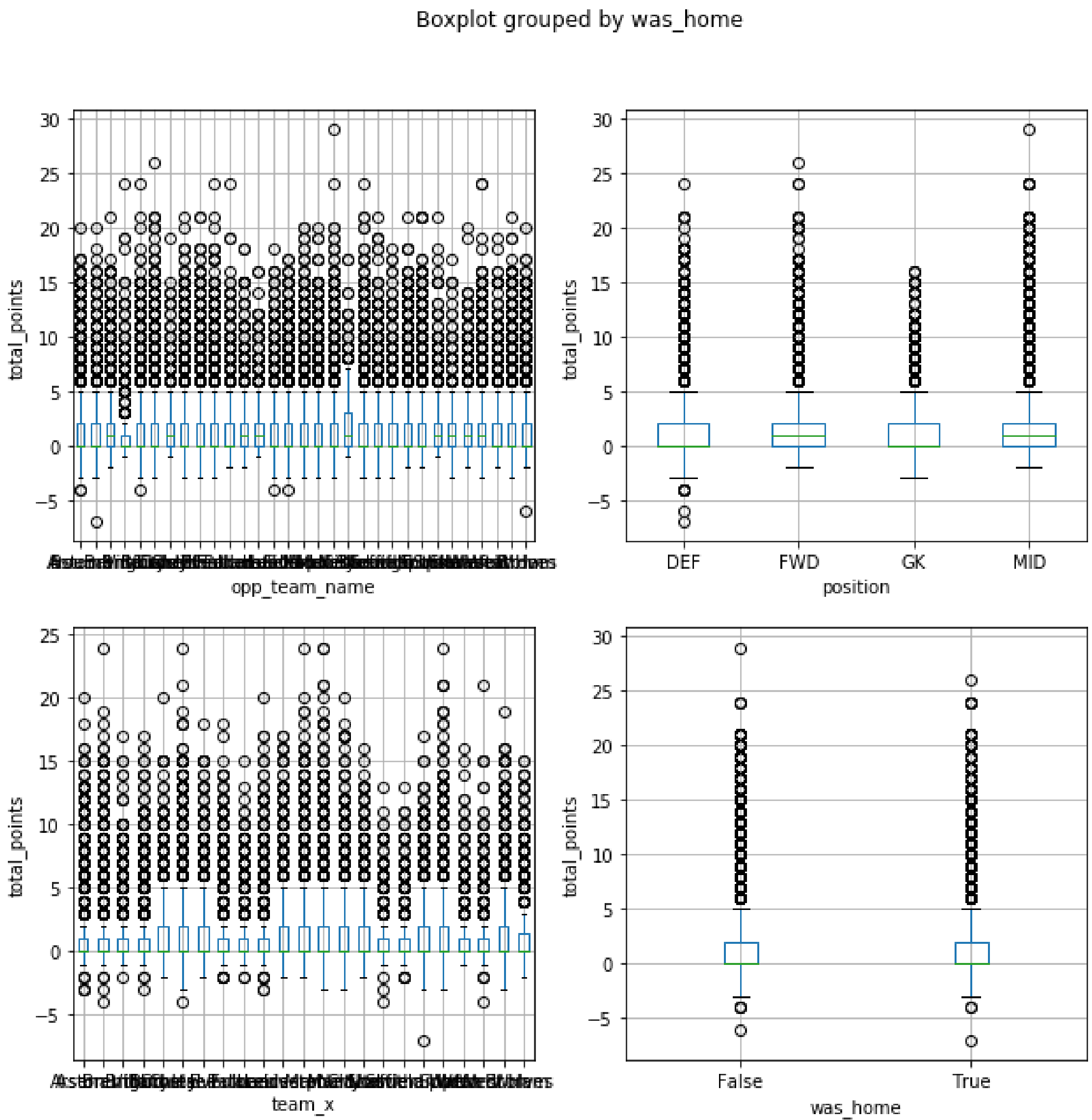
[122]:
```
# Examine the correlation with the target for numerical categorical features
 ↪(floats with <10 unique values)

fig, axes = plt.subplots(3, 3, figsize=(20,20))

for index, feature in enumerate(sorted(num_cat_vars)):
    x_subplot = index//3
    y_subplot = index%3
    boxplt = df.boxplot(column=['total_points'], by=[feature],
 ↪ax=axes[x_subplot, y_subplot])
```

```
    boxplt.set_xlabel(feature)
    boxplt.set_ylabel('total_points')
    boxplt.set_title('')
```



[125]:
```
# Examine the correlation with the target for categorical features

fig, axes = plt.subplots(2, 2, figsize=(10,10))

for index, feature in enumerate(sorted(cat_vars)):
    x_subplot = index//2
    y_subplot = index%2
```

```
    boxplt = df.boxplot(column=['total_points'], by=[feature],␣
↪ax=axes[x_subplot, y_subplot])
    boxplt.set_xlabel(feature)
    boxplt.set_ylabel('total_points')
    boxplt.set_title('')
```



Let's look at highly correlated features

[128]:
```
corr_matrix = df_x.corr()
corr_matrix.style.background_gradient(cmap='coolwarm')
```

[128]: <pandas.io.formats.style.Styler at 0x7fd1a22c1690>

[137]:

```python
# Take a loot at the most highly correlated features

df_x.corr().abs().unstack().sort_values(kind='quicksort')[-41:-31]
```

[137]:

```
threat     ict_index    0.838204
ict_index  threat       0.838204
           influence    0.838599
influence  ict_index    0.838599
           bps          0.901690
bps        influence    0.901690
fixture    round        0.977056
           GW           0.977056
GW         fixture      0.977056
round      fixture      0.977056
dtype: float64
```