Multivariate Analysis

Classification

In [42]:
```
import pandas as pd
import numpy as np
```

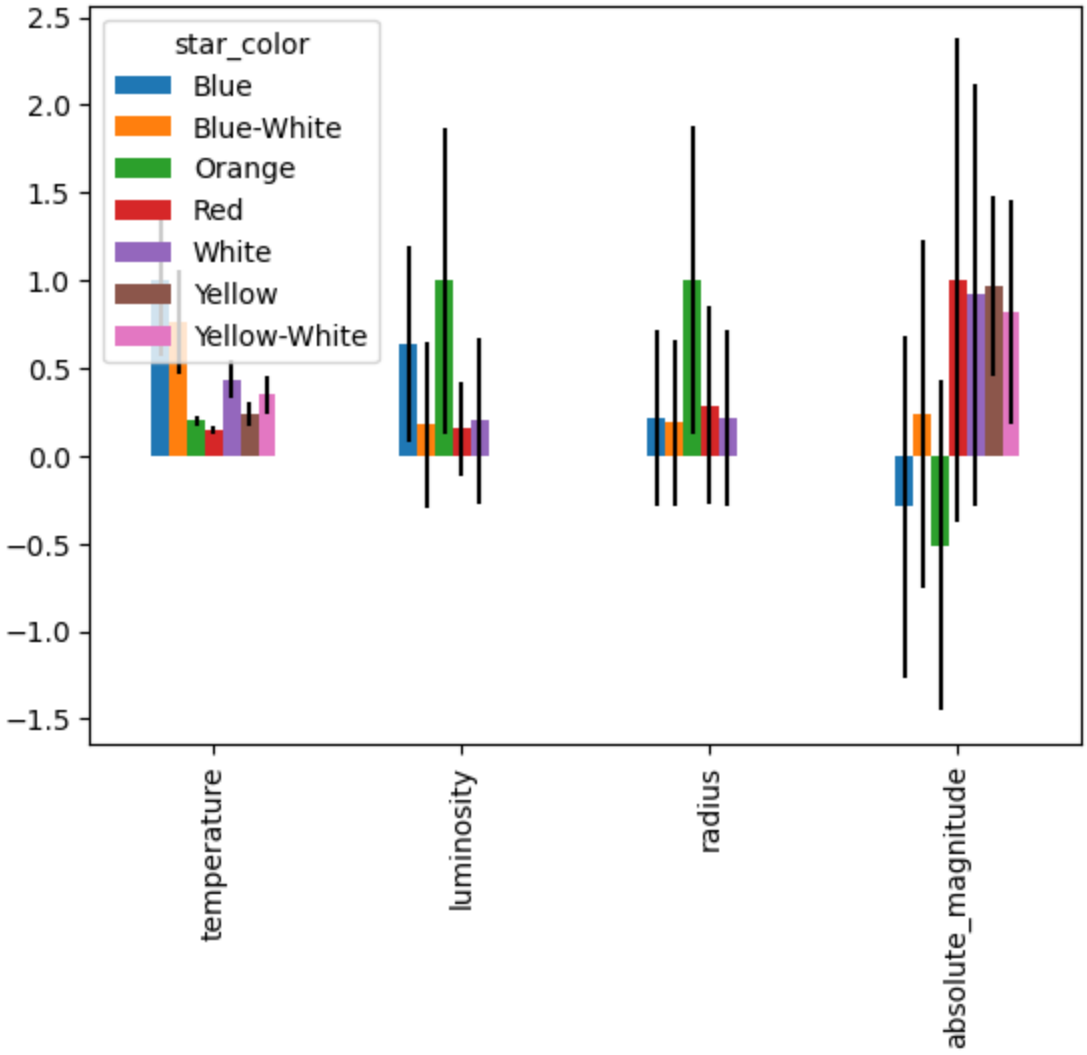
In [43]:
```
data = pd.read_csv('star_dataset.csv')
data.rename(columns={'Temperature (K)': 'temperature', 'Luminosity(L/Lo)' :'luminosity',
data.head()
```

Out[43]:

	temperature	luminosity	radius	absolute_magnitude	star_type	star_color	spectral_class
0	3068	0.002400	0.1700	16.12	Red Dwarf	Red	M
1	3042	0.000500	0.1542	16.60	Red Dwarf	Red	M
2	2600	0.000300	0.1020	18.70	Red Dwarf	Red	M
3	2800	0.000200	0.1600	16.65	Red Dwarf	Red	M
4	1939	0.000138	0.1030	20.06	Red Dwarf	Red	M

In [44]:
```
grouped = data.groupby('star_color')
normalized = (grouped.mean() / grouped.mean().max()).transpose()
errors = (grouped.std() / grouped.mean().max()).transpose()
normalized.plot(kind='bar', yerr=errors)
```

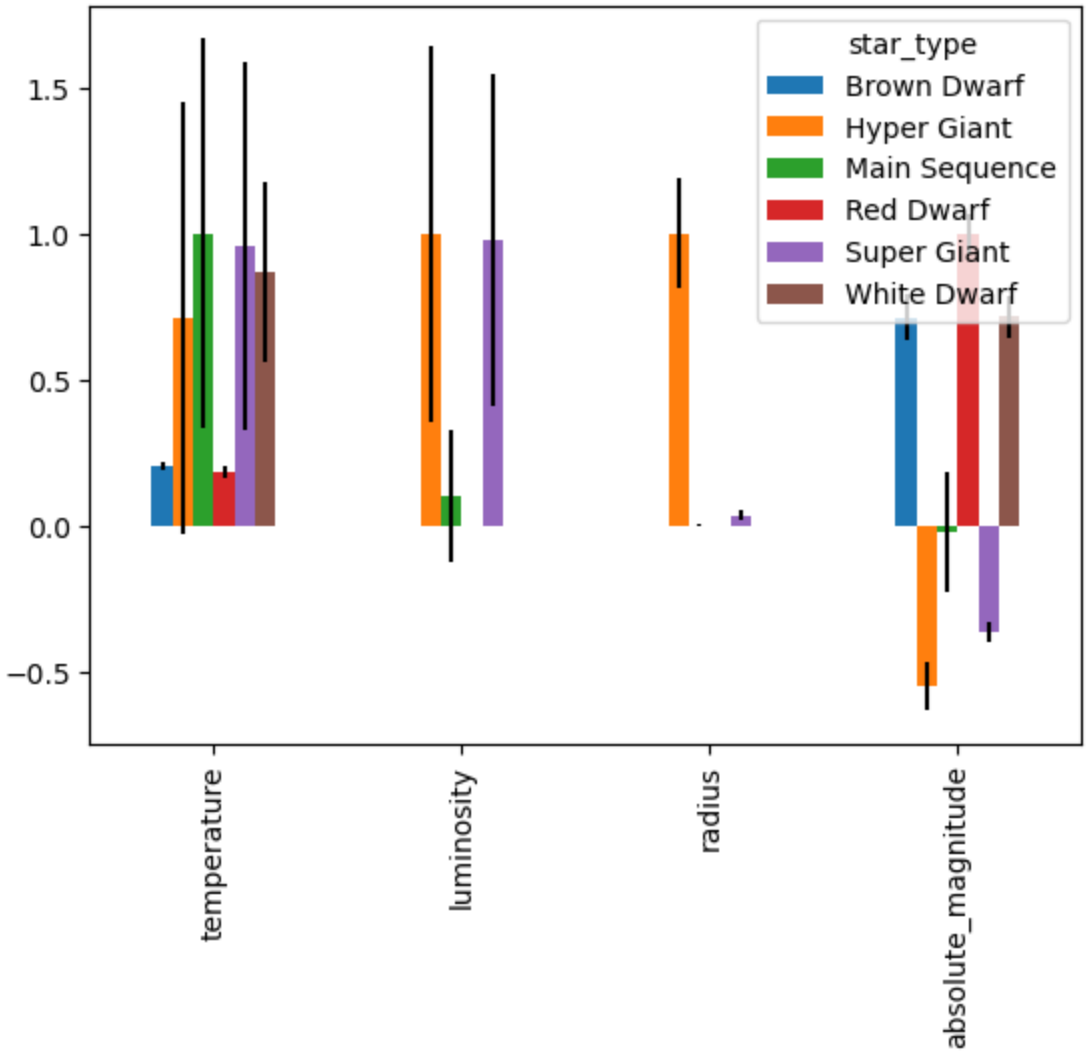
Out[44]:
```
<AxesSubplot:>
```



Het lijkt hier dat star color alleen goed kan worden bepaald door de temperatuur, dus is dit geen handige categorie om te kiezen voor de multivariable analysis. Spectral type had een leuke categorie geweest om te voorspellen, echter spectral type is gerelateerd aan de kleur en er is maar één waarde voor G, waardoor de analyse niet kan worden uitgevoerd tenzij deze waarde eruit wordt gehaald.

In [45]:
```
grouped = data.groupby('star_type')
normalized = (grouped.mean() / grouped.mean().max()).transpose()
errors = (grouped.std() / grouped.mean().max()).transpose()
normalized.plot(kind='bar', yerr=errors)
```

Out[45]:
```
<AxesSubplot:>
```



Star type heeft wel significante verschillen bij de radius (zoals ook te zien in assignment 12) en de absolute magnitude.

In [46]:
```
from sklearn.model_selection import train_test_split
```

In [47]:
```
star_type = 'star_type'
data_train_st, data_test_st = train_test_split(data, test_size=0.3, random_state=42, str
```

In [48]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [49]:
```
properties_st = ['radius', 'absolute_magnitude']
dt_classification_st = DecisionTreeClassifier(max_depth = 10)
dt_classification_st.fit(data_train_st[properties_st], data_train_st[star_type])
```

Out[49]:
```
DecisionTreeClassifier(max_depth=10)
```

In [50]:
```
def calculate_accuracy(predictions, actuals):
    if(len(predictions) != len(actuals)):
        raise Exception("The amount of predictions did not equal the amount of actuals")

    return (predictions == actuals).sum() / len(actuals)
```

In [51]:
```
predictionsOnTrainset_st = dt_classification_st.predict(data_train_st[properties_st])
predictionsOnTestset_st = dt_classification_st.predict(data_test_st[properties_st])

accuracyTrain_st = calculate_accuracy(predictionsOnTrainset_st, data_train_st.star_type)
accuracyTest_st = calculate_accuracy(predictionsOnTestset_st, data_test_st.star_type)
```

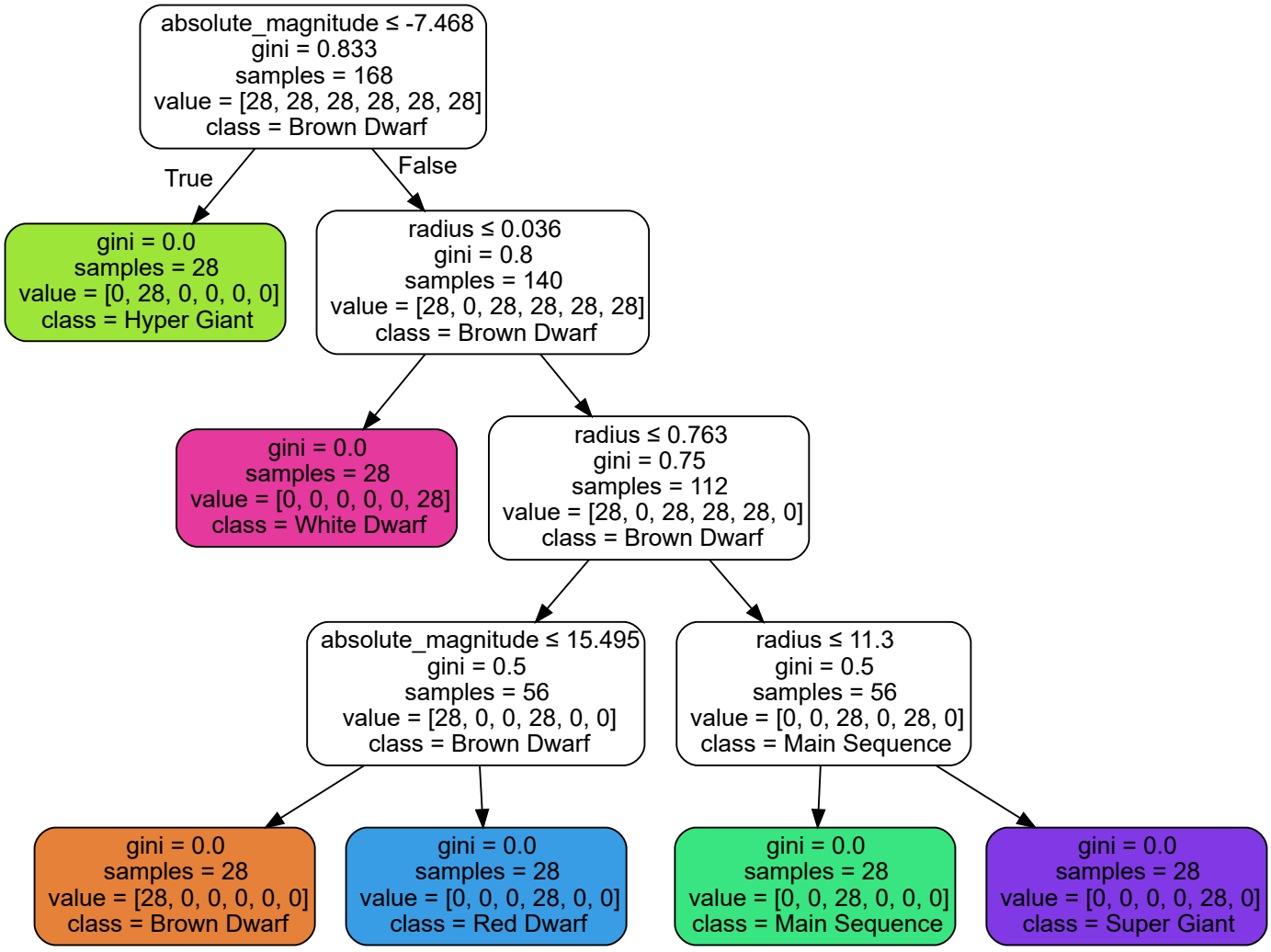
```
print("Accuracy on training set " + str(accuracyTrain_st))
print("Accuracy on test set " + str(accuracyTest_st))
```

```
Accuracy on training set 1.0
Accuracy on test set 1.0
```

In [52]:
```
from sklearn import tree
import graphviz

def plot_tree_classification(model, features, class_names):
    # Generate plot data
    dot_data = tree.export_graphviz(model, out_file=None,
                                    feature_names=features,
                                    class_names=class_names,
                                    filled=True, rounded=True,
                                    special_characters=True)

    # Turn into graph using graphviz
    graph = graphviz.Source(dot_data)

    # Write out a pdf
    graph.render("Trees/decision_tree_16")

    # Display in the notebook
    return graph
```

In [53]:
```
plot_tree_classification(dt_classification_st, properties_st, np.sort(data.star_type.uni
```

Out[53]:



De accuracy is erg hoog voor beide de training en de test set. Ook heeft de decision tree maar een diepte van 4 nodig om deze accuracy tot stand te brengen.

We kunnen ook nog kijken naar kleur, om te zien of het inderdaad klopt dat je moeilijk de kleur kan voorspellen met radius en absolute magnitude.

In [54]:
```
star_color = 'star_color'
data_train_sc, data_test_sc = train_test_split(data, test_size=0.3, random_state=42, str
dt_classification_sc = DecisionTreeClassifier(max_depth = 10)
dt_classification_sc.fit(data_train_sc[properties_st], data_train_sc[star_color])
```

Out[54]:
```
DecisionTreeClassifier(max_depth=10)
```

In [55]:
```
predictionsOnTrainset_sc = dt_classification_sc.predict(data_train_sc[properties_st])
predictionsOnTestset_sc = dt_classification_sc.predict(data_test_sc[properties_st])

accuracyTrain_sc = calculate_accuracy(predictionsOnTrainset_sc, data_train_sc.star_color
accuracyTest_sc = calculate_accuracy(predictionsOnTestset_sc, data_test_sc.star_color)

print("Accuracy on training set " + str(accuracyTrain_sc))
print("Accuracy on test set " + str(accuracyTest_sc))
```

```
Accuracy on training set 0.9940476190476191
Accuracy on test set 0.6666666666666666
```

Je kan zien dat de accuracy voor de test set een stuk lager is wanneer star color wordt voorspeld met dezelfde kolommen, hier kan dus geen goede fit voor gemaakt worden.