In [9]:
```
import numpy as np
import pandas as pd
%matplotlib inline

import random
from pprint import pprint

from decision_tree_functions import decision_tree_algorithm, decision_tree_predictions
from helper_functions import train_test_split, calculate_accuracy
```

Tải và chuẩn bị dữ liệu

Định dạng dữ liệu

- Cột cuối cùng của khung dữ liệu phải chứa nhãn và nó cũng phải được gọi là "nhãn" "label"
- Không được có giá trị nào bị thiếu trong khung dữ liệu

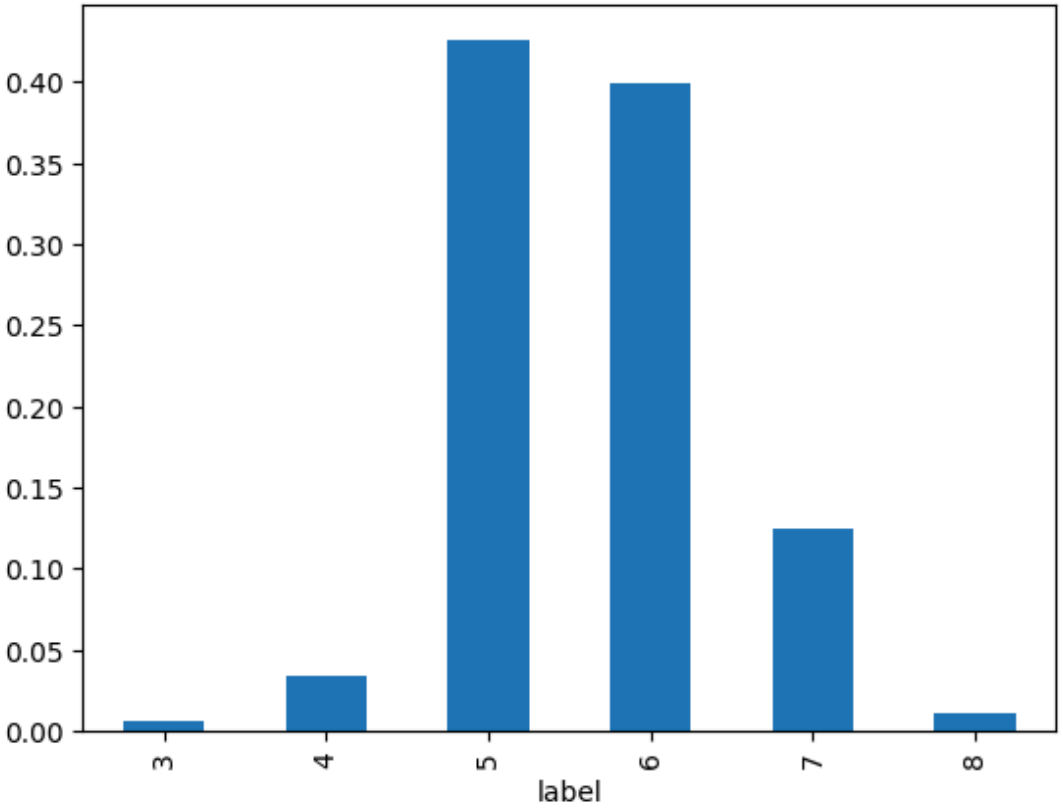
In [10]:
```
df = pd.read_csv("./data/winequality-red.csv")
df["label"] = df.quality
df = df.drop("quality", axis=1)

column_names = []
for column in df.columns:
    name = column.replace(" ", "_")
    column_names.append(name)
df.columns = column_names

df.head()
```

Out[10]:

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulpha
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0

In [11]:
```
wine_quality = df.label.value_counts(normalize=True)
wine_quality = wine_quality.sort_index()
wine_quality.plot(kind="bar")
```

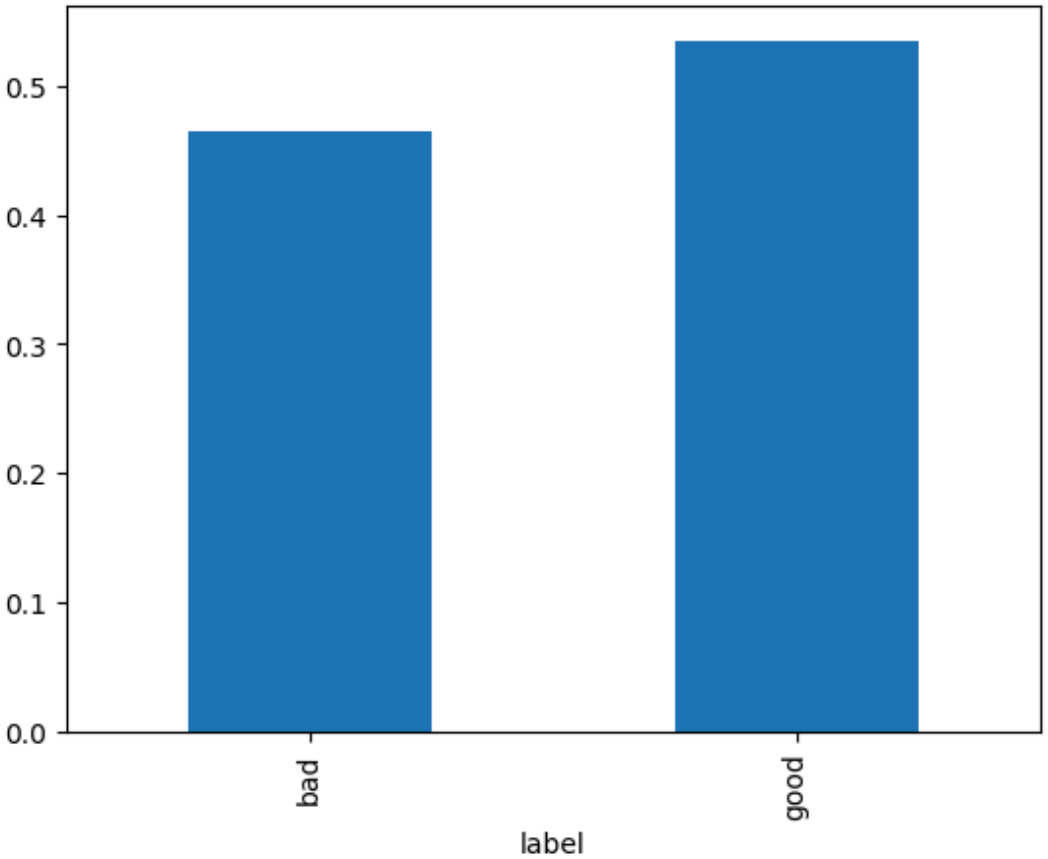
Out[11]:
```
<Axes: xlabel='label'>
```



In [12]:
```python
def transform_label(value):
    if value <= 5:
        return "bad"
    else:
        return "good"

df["label"] = df.label.apply(transform_label)
```

In [13]:
```python
wine_quality = df.label.value_counts(normalize=True)
wine_quality[["bad", "good"]].plot(kind="bar")
wine_quality
```

Out[13]:
```
label
good    0.534709
bad     0.465291
Name: proportion, dtype: float64
```



In [14]:
```
random.seed(0)
train_df, test_df = train_test_split(df, test_size=0.2)
```

Random Forest

In [15]:
```
def bootstrapping(train_df, n_bootstrap):
    bootstrap_indices = np.random.randint(low=0, high=len(train_df), size=n_bootstrap)
```

```
    df_bootstrapped = train_df.iloc[bootstrap_indices]

    return df_bootstrapped

def random_forest_algorithm(train_df, n_trees, n_bootstrap, n_features, dt_max_depth):
    forest = []
    for i in range(n_trees):
        df_bootstrapped = bootstrapping(train_df, n_bootstrap)
        tree = decision_tree_algorithm(df_bootstrapped, max_depth=dt_max_depth, random_subspace=n_features)
        forest.append(tree)

    return forest

def random_forest_predictions(test_df, forest):
    df_predictions = {}
    for i in range(len(forest)):
        column_name = "tree_{}".format(i)
        predictions = decision_tree_predictions(test_df, tree=forest[i])
        df_predictions[column_name] = predictions

    df_predictions = pd.DataFrame(df_predictions)
    random_forest_predictions = df_predictions.mode(axis=1)[0]

    return random_forest_predictions
```

In [18]:

```
forest = random_forest_algorithm(train_df, n_trees=4, n_bootstrap=800, n_features=2, dt_max_depth=4)
predictions = random_forest_predictions(test_df, forest)
accuracy = calculate_accuracy(predictions, test_df.label)

print("Accuracy = {}".format(accuracy))
```

```
Accuracy = 0.734375
```