```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
```

```
In [2]: df= pd.read_csv("C:/Users/doppa/Downloads/insurance.csv")
        df.head()
```

Out[2]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

```
In [3]: df.shape
Out[3]: (1338, 7)
```

```
In [4]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1338 non-null   int64  
 1   sex       1338 non-null   object 
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64  
 4   smoker    1338 non-null   object 
 5   region    1338 non-null   object 
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

```
In [5]: df.isnull()
```

Out[5]:

	age	sex	bmi	children	smoker	region	charges
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
1333	False	False	False	False	False	False	False
1334	False	False	False	False	False	False	False
1335	False	False	False	False	False	False	False
1336	False	False	False	False	False	False	False
1337	False	False	False	False	False	False	False

1338 rows × 7 columns

```
In [6]: df.duplicated()

Out[6]:
0       False
1       False
2       False
3       False
4       False
        ...  
1333    False
1334    False
1335    False
1336    False
1337    False
Length: 1338, dtype: bool
```

Data Preprocessing

split the data

```
In [7]: y = df.pop("charges")
        X = df
```

```
In [8]: from sklearn.model_selection import train_test_split

        X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                            test_size = 0.25,
                                                            random_state = 0)

        print(X_train.shape, y_train.shape)
        print(y_test.shape, y_test.shape)

(1003, 6) (1003,)
(335,) (335,)
```

Separate Numerical and Categorical Features¶

```
In [9]: X_train_cat = X_train.select_dtypes(include=['object'])
        X_train_num = X_train.select_dtypes(include=['int64', 'float64'])
```

```
In [10]: # Rescaling numerical features
         from sklearn.preprocessing import StandardScaler

         scaler = StandardScaler()

         # column names are (annoyingly) lost after Scaling
         # (i.e. the dataframe is converted to a numpy ndarray)
         X_train_num_transformed = pd.DataFrame(scaler.fit_transform(X_train_num),
                                                columns = scaler.get_feature_names_out(),
                                                index = X_train_num.index)

         X_train_num_transformed.head()
```

Out[10]:

	age	bmi	children
1075	-0.514853	-0.181331	-0.063607
131	1.548746	-1.393130	-0.892144
15	-1.439915	-0.982242	-0.063607
1223	-1.368757	-1.011133	-0.892144
1137	-0.941805	-1.362635	-0.892144

Testing

```
In [11]: X_test_cat = X_test.select_dtypes(include=['object'])
         X_test_num = X_test.select_dtypes(include=['int64', 'float64'])
```

```
In [12]: X_test_num_transformed = pd.DataFrame(scaler.transform(X_test_num),
                                               columns=X_test_num.columns,
                                               index=X_test_num.index)
         X_test_num_transformed.head()
```

Out[12]:

	age	bmi	children
578	0.908319	-0.083424	-0.063607
610	0.552526	-0.216642	-0.063607
569	0.623684	1.580192	0.764931
1034	1.548746	1.229492	-0.892144
198	0.837160	-2.033538	-0.892144

Feature Engineering: Applying One-Hot Encoding on Categorical Features

```
In [13]: from sklearn.preprocessing import OneHotEncoder

         encoder_ = OneHotEncoder(sparse_output=False)

         X_train_cat_tansformed = pd.DataFrame(encoder_.fit_transform(X_train_cat),
                                     columns=encoder_.get_feature_names_out(),
                                     index = X_train_cat.index)

         print("Shape of Data before Transformation:", X_train_cat.shape)
         print("Shape of Data after Transformation:", X_train_cat_tansformed.shape)

         X_train_cat_tansformed.head()

Shape of Data before Transformation: (1003, 3)
Shape of Data after Transformation: (1003, 8)
```

Out[13]:

	sex_female	sex_male	smoker_no	smoker_yes	region_northeast	region_northwest	region_southeast	region_southwest
1075	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0
131	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0
15	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0
1223	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0
1137	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0

Testing

```
In [14]: X_test_cat = X_test.select_dtypes(include=['object'])
         X_test_num = X_test.select_dtypes(include=['int64', 'float64'])
```

```
In [15]: X_test_cat_tansformed = pd.DataFrame(encoder_.transform(X_test_cat),
                                  columns=encoder_.get_feature_names_out(),
                                  index=X_test_cat.index)
         X_test_cat_tansformed.head()
```

Out[15]:

	sex_female	sex_male	smoker_no	smoker_yes	region_northeast	region_northwest	region_southeast	region_southwest
578	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0
610	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0
569	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0
1034	0.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0
198	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0

```
In [16]: X_train_transformed = pd.concat([X_train_num_transformed, X_train_cat_tansformed], axis=1)
         X_train_transformed.head()
```

Out[16]:

	age	bmi	children	sex_female	sex_male	smoker_no	smoker_yes	region_northeast	region_northwest	region_southeast	region_southwest
1075	-0.514853	-0.181331	-0.063607	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0
131	1.548746	-1.393130	-0.892144	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0
15	-1.439915	-0.982242	-0.063607	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0
1223	-1.368757	-1.011133	-0.892144	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0
1137	-0.941805	-1.362635	-0.892144	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0

X_test_transformed

```
In [17]: X_test_transformed = pd.concat([X_test_num_transformed, X_test_cat_tansformed], axis=1)
         X_test_transformed.head()
```

Out[17]:

	age	bmi	children	sex_female	sex_male	smoker_no	smoker_yes	region_northeast	region_northwest	region_southeast	region_southwest
578	0.908319	-0.083424	-0.063607	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0
610	0.552526	-0.216642	-0.063607	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0
569	0.623684	1.580192	0.764931	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0
1034	1.548746	1.229492	-0.892144	0.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0
198	0.837160	-2.033538	-0.892144	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0

Model Training

KNN

```
In [18]: from sklearn.neighbors import KNeighborsRegressor
         regressor = KNeighborsRegressor()
         regressor.fit(X_train_transformed, y_train)

Out[18]: ▾ KNeighborsRegressor
         KNeighborsRegressor()
```

```
In [19]: y_test_pred = regressor.predict(X_test_transformed)
```

```
In [20]: from sklearn import metrics
         metrics.r2_score(y_test, y_test_pred)

Out[20]: 0.8067310484954536
```

```
In [21]: output_df = pd.DataFrame({'Actual': y_test.values.flatten()}, index=X_test.index)
```

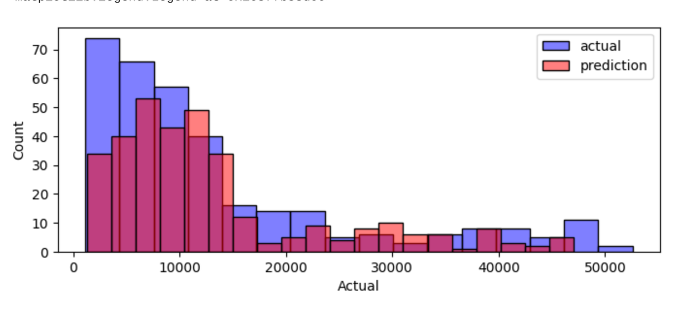
```
In [22]: output_df['KNN Regression Predictions'] = y_test_pred

         output_df
```

Out[22]:

	Actual	KNN Regression Predictions
578	9724.53000	8433.604400
610	8547.69130	7948.160220
569	45702.02235	40597.527604
1034	12950.07120	11324.034480
198	9644.25250	11441.757950
...	...	...
574	13224.05705	16827.301782
1174	4433.91590	10148.848410
1327	9377.90470	13396.970186
817	3597.59600	5580.541200
1337	29141.36030	30285.366580

335 rows × 2 columns

```
In [23]: fig, ax = plt.subplots(figsize=(8,3))

         sns.histplot(output_df['Actual'], color='blue', alpha=0.5, label="actual")
         sns.histplot(output_df['KNN Regression Predictions'], color='red', alpha=0.5, label="prediction")

         plt.legend()

Out[23]: <matplotlib.legend.Legend at 0x16877b58d00>
```



Decisison Tree

```
In [24]: from sklearn.tree import DecisionTreeRegressor
         regressor = DecisionTreeRegressor()
         regressor.fit(X_train_transformed, y_train)

Out[24]: ▾ DecisionTreeRegressor
         DecisionTreeRegressor()
```

```
In [25]: from sklearn import metrics
         metrics.r2_score(y_test, y_test_pred)

Out[25]: 0.8067310484954536
```

```
In [26]: y_test_pred = regressor.predict(X_test_transformed)
```

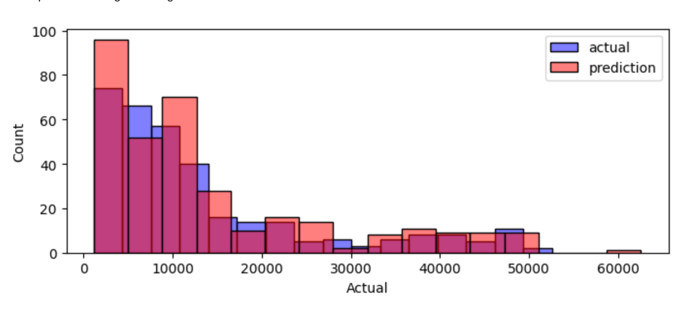
```
In [27]: output_df['DT Regression Predictions'] = y_test_pred

         output_df
```

Out[27]:

	Actual	KNN Regression Predictions	DT Regression Predictions
578	9724.53000	8433.604400	10085.8460
610	8547.69130	7948.160220	8233.0975
569	45702.02235	40597.527604	44202.6536
1034	12950.07120	11324.034480	13429.0354
198	9644.25250	11441.757950	9855.1314
...	...	...	...
574	13224.05705	16827.301782	13430.2650
1174	4433.91590	10148.848410	4922.9159
1327	9377.90470	13396.970186	9872.7010
817	3597.59600	5580.541200	3443.0640
1337	29141.36030	30285.366580	29523.1656

335 rows × 3 columns

```
In [28]: fig, ax = plt.subplots(figsize=(8,3))

         sns.histplot(output_df['Actual'], color='blue', alpha=0.5, label="actual")
         sns.histplot(output_df['DT Regression Predictions'], color='red', alpha=0.5, label="prediction")

         plt.legend()

Out[28]: <matplotlib.legend.Legend at 0x16879cb9990>
```



Linear Regression

```
In [29]: from sklearn.linear_model import LinearRegression
         regressor = LinearRegression()
         regressor.fit(X_train_transformed, y_train)

Out[29]: ▾ LinearRegression
         LinearRegression()
```

```
In [30]: y_test_pred = regressor.predict(X_test_transformed)
```

```
In [31]: from sklearn import metrics
         metrics.r2_score(y_test, y_test_pred)

Out[31]: 0.7958786376014413
```

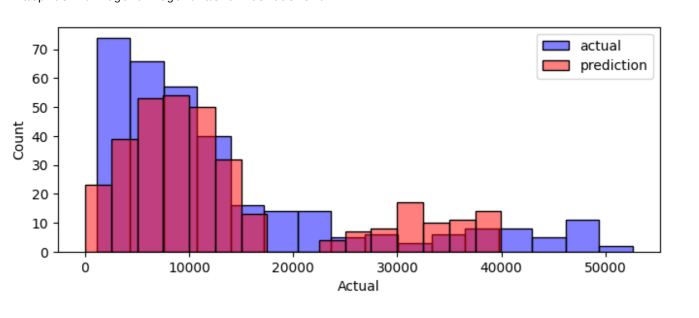
```
In [32]: output_df['Linear Regression Predictions'] = y_test_pred

         output_df
```

Out[32]:

	Actual	KNN Regression Predictions	DT Regression Predictions	Linear Regression Predictions
578	9724.53000	8433.604400	10085.8460	11121.101409
610	8547.69130	7948.160220	8233.0975	9369.083452
569	45702.02235	40597.527604	44202.6536	38349.258807
1034	12950.07120	11324.034480	13429.0354	16331.935006
198	9644.25250	11441.757950	9855.1314	7041.227706
...	...	...	...	...
574	13224.05705	16827.301782	13430.2650	15034.106647
1174	4433.91590	10148.848410	4922.9159	7121.723799
1327	9377.90470	13396.970186	9872.7010	10602.930893
817	3597.59600	5580.541200	3443.0640	7050.864796
1337	29141.36030	30285.366580	29523.1656	36872.299727

335 rows × 4 columns

```
In [33]: fig, ax = plt.subplots(figsize=(8,3))

         sns.histplot(output_df['Actual'], color='blue', alpha=0.5, label="actual")
         sns.histplot(output_df['Linear Regression Predictions'], color='red', alpha=0.5, label="prediction")

         plt.legend()

Out[33]: <matplotlib.legend.Legend at 0x16879d8f820>
```



Random Forest

```
In [34]: from sklearn.ensemble import RandomForestRegressor
         regressor = RandomForestRegressor()
         regressor.fit(X_train_transformed, y_train)

Out[34]: ▾ RandomForestRegressor
         RandomForestRegressor()
```

```
In [35]: y_test_pred = regressor.predict(X_test_transformed)
```

```
In [36]: from sklearn import metrics
         metrics.r2_score(y_test, y_test_pred)

Out[36]: 0.8733772998555664
```

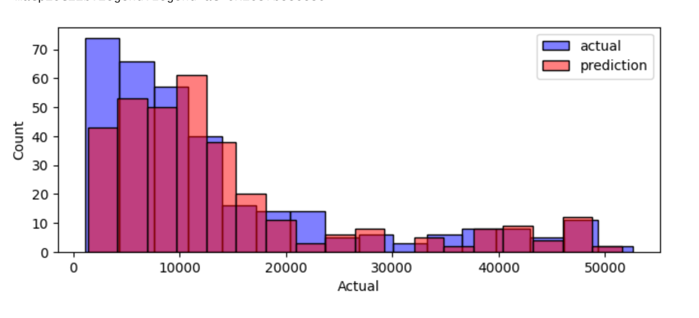
```
In [37]: output_df['RF Regression Predictions'] = y_test_pred

         output_df
```

Out[37]:

	Actual	KNN Regression Predictions	DT Regression Predictions	Linear Regression Predictions	RF Regression Predictions
578	9724.53000	8433.604400	10085.8460	11121.101409	10748.164891
610	8547.69130	7948.160220	8233.0975	9369.083452	9223.268982
569	45702.02235	40597.527604	44202.6536	38349.258807	45554.203310
1034	12950.07120	11324.034480	13429.0354	16331.935006	13044.096473
198	9644.25250	11441.757950	9855.1314	7041.227706	9430.215642
...	...	...	...	...	...
574	13224.05705	16827.301782	13430.2650	15034.106647	17198.019827
1174	4433.91590	10148.848410	4922.9159	7121.723799	11746.910435
1327	9377.90470	13396.970186	9872.7010	10602.930893	10316.203067
817	3597.59600	5580.541200	3443.0640	7050.864796	4464.031106
1337	29141.36030	30285.366580	29523.1656	36872.299727	28823.706828

335 rows × 5 columns

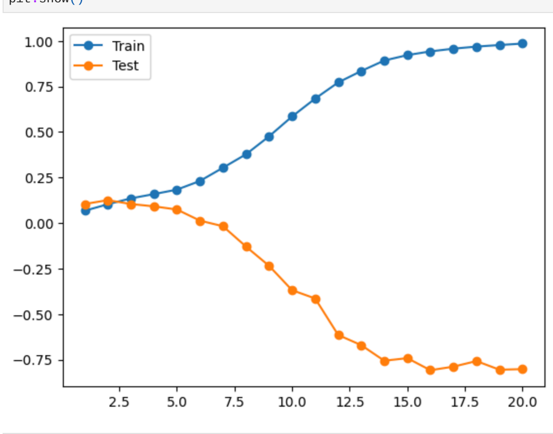
```
In [38]: fig, ax = plt.subplots(figsize=(8,3))

         sns.histplot(output_df['Actual'], color='blue', alpha=0.5, label="actual")
         sns.histplot(output_df['RF Regression Predictions'], color='red', alpha=0.5, label="prediction")

         plt.legend()

Out[38]: <matplotlib.legend.Legend at 0x1687bee6c50>
```



Hyperparameter tuning

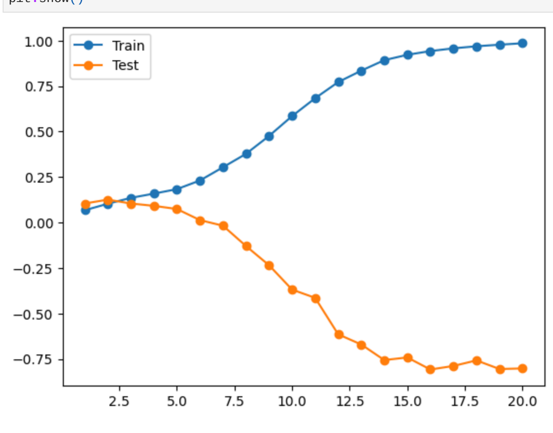
```
In [48]: from sklearn.neighbors import KNeighborsRegressor
         from sklearn.metrics import r2_score

         train_scores, test_scores = list(), list()

         values = [i for i in range(1, 21)]

         for i in values:
             model = KNeighborsRegressor(n_neighbors=i)

             model.fit(X_train_num_transformed, y_train)

             y_train_pred = model.predict(X_train_num_transformed)
             train_score = r2_score(y_train, y_train_pred)
             train_scores.append(train_score)

             y_test_pred = model.predict(X_test_num_transformed)
             test_score = r2_score(y_test, y_test_pred)
             test_scores.append(test_score)

             print('> %d, train: %.3f, test: %.3f' % (i, train_score, test_score))

> 1, train: 0.984, test: -0.892
> 2, train: 0.545, test: -0.399
> 3, train: 0.390, test: -0.225
> 4, train: 0.327, test: -0.079
> 5, train: 0.260, test: -0.056
> 6, train: 0.224, test: 0.006
> 7, train: 0.211, test: 0.030
> 8, train: 0.207, test: 0.045
> 9, train: 0.195, test: 0.040
> 10, train: 0.187, test: 0.024
> 11, train: 0.186, test: 0.041
> 12, train: 0.181, test: 0.057
> 13, train: 0.173, test: 0.059
> 14, train: 0.169, test: 0.068
> 15, train: 0.165, test: 0.068
> 16, train: 0.162, test: 0.071
> 17, train: 0.160, test: 0.069
> 18, train: 0.155, test: 0.084
> 19, train: 0.153, test: 0.093
> 20, train: 0.150, test: 0.095
```

```
In [53]: # plot of train and test scores vs tree depth
         plt.plot(values, train_scores, '-o', label='Train')
         plt.plot(values, test_scores, '-o', label='Test')
         plt.legend()
         plt.show()
```




```
In [51]: from sklearn.tree import DecisionTreeRegressor
         from sklearn.metrics import r2_score

         train_scores, test_scores = list(), list()

         values = [i for i in range(1, 21)]

         for i in values:
             model = DecisionTreeRegressor(max_depth=i)

             model.fit(X_train_num_transformed, y_train)

             y_train_pred = model.predict(X_train_num_transformed)
             train_score = r2_score(y_train, y_train_pred)
             train_scores.append(train_score)

             y_test_pred = model.predict(X_test_num_transformed)
             test_score = r2_score(y_test, y_test_pred)
             test_scores.append(test_score)

             print('> %d, train: %.3f, test: %.3f' % (i, train_score, test_score))

> 1, train: 0.067, test: 0.104
> 2, train: 0.103, test: 0.126
> 3, train: 0.135, test: 0.104
> 4, train: 0.159, test: 0.092
> 5, train: 0.183, test: 0.074
> 6, train: 0.231, test: 0.013
> 7, train: 0.304, test: -0.018
> 8, train: 0.377, test: -0.129
> 9, train: 0.476, test: -0.235
> 10, train: 0.585, test: -0.369
> 11, train: 0.683, test: -0.414
> 12, train: 0.772, test: -0.614
> 13, train: 0.834, test: -0.670
> 14, train: 0.892, test: -0.756
> 15, train: 0.922, test: -0.742
> 16, train: 0.942, test: -0.808
> 17, train: 0.957, test: -0.788
> 18, train: 0.968, test: -0.758
> 19, train: 0.977, test: -0.805
> 20, train: 0.985, test: -0.802
```

```
In [52]: # plot of train and test scores vs tree depth
         plt.plot(values, train_scores, '-o', label='Train')
         plt.plot(values, test_scores, '-o', label='Test')
         plt.legend()
         plt.show()
```



Cross Validation

GridSearchCV

```
In [54]: from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
         from sklearn.metrics import classification_report
```

```
In [55]: from sklearn.neighbors import KNeighborsRegressor
```

```
In [60]: tuned_parameters = [{'n_neighbors':[i for i in range(1, 51)], 'p':[1, 2, 3]}]

         clf = GridSearchCV(
             estimator=KNeighborsRegressor(),
             param_grid=tuned_parameters,
             scoring='r2',
             cv=5,
             return_train_score=True,
             verbose=1
         )
         clf.fit(X_train_num_transformed, y_train)

Fitting 5 folds for each of 150 candidates, totalling 750 fits
Out[60]: ▸ GridSearchCV
         ▸ estimator: KNeighborsRegressor
           ▸ KNeighborsRegressor
```

```
In [63]: print("Best parameters set found on train set")
         print(clf.best_params_)
         print(clf.best_estimator_)
         print()

         print('Score on Test Data: ', clf.score(X_test_num_transformed, y_test))

Best parameters set found on train set
{'n_neighbors': 50, 'p': 2}
KNeighborsRegressor(n_neighbors=50)

Score on Test Data:  0.14436959367079105
```

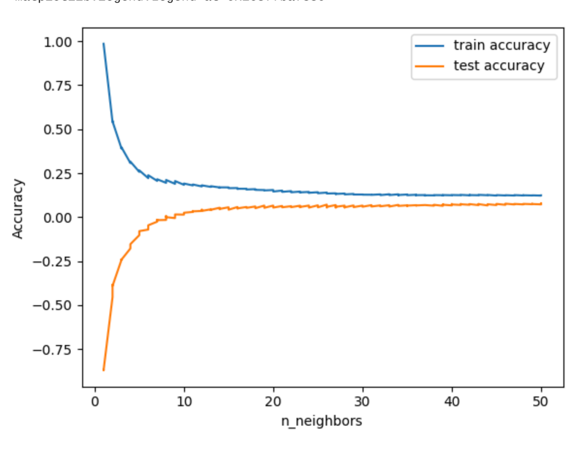
```
In [64]: cv_results = pd.DataFrame(clf.cv_results_)

         cv_results.head()
```

Out[64]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_n_neighbors	param_p	params	split0_test_score	split1_test_score	split2_test_score	...	mean_test_score	std_test_score	rank_test_score	split0_train
0	0.002693	0.002519	0.010307	0.005910	1	1	{'n_neighbors': 1, 'p': 1}	-0.830206	-0.982420	-0.982717	...	-0.866922	0.102458	149	0.
1	0.015322	0.009919	0.007296	0.012217	1	2	{'n_neighbors': 1, 'p': 2}	-0.802040	-1.087539	-0.908380	...	-0.865475	0.137050	148	0.
2	0.013647	0.013596	0.015932	0.024303	1	3	{'n_neighbors': 1, 'p': 3}	-0.846500	-1.030832	-0.900823	...	-0.869234	0.126456	150	0.
3	0.008203	0.012127	0.000220	0.000441	2	1	{'n_neighbors': 2, 'p': 1}	-0.400038	-0.550681	-0.506832	...	-0.454379	0.063415	147	0.
4	0.009072	0.007429	0.009742	0.007982	2	2	{'n_neighbors': 2, 'p': 2}	-0.361025	-0.508534	-0.380275	...	-0.385539	0.067276	145	0.

5 rows × 22 columns

```
In [65]: plt.plot(cv_results['param_n_neighbors'], cv_results['mean_train_score'])
         plt.plot(cv_results['param_n_neighbors'], cv_results['mean_test_score'])
         plt.xlabel('n_neighbors')
         plt.ylabel('Accuracy')
         plt.legend(['train accuracy', 'test accuracy'], loc='upper right')

Out[65]: <matplotlib.legend.Legend at 0x1687fba7880>
```



RandomizedSearchCV

```
In [70]: tuned_parameters = [{'n_neighbors': [i for i in range(1, 51)],
                              'p': [1, 2, 3]}]

         clf = RandomizedSearchCV(
             estimator=KNeighborsRegressor(),
             param_distributions=tuned_parameters,
             scoring='r2',
             cv=5,
             return_train_score=True,
             verbose=1
         )
         clf.fit(X_train_num_transformed, y_train)

Fitting 5 folds for each of 10 candidates, totalling 50 fits
Out[70]: ▸ RandomizedSearchCV
         ▸ estimator: KNeighborsRegressor
           ▸ KNeighborsRegressor
```

```
In [72]: print("Best parameters set found on train set:")
         print(clf.best_params_)
         print(clf.best_estimator_)
         print()

         print('Score on Test Data: ', clf.score(X_test_num_transformed, y_test))

Best parameters set found on train set:
{'p': 2, 'n_neighbors': 41}
KNeighborsRegressor(n_neighbors=41)

Score on Test Data:  0.13550313380793444
```

```
In [ ]:
```