PD3

Artur Żółkowski

In [43]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, KFold, cross_validate
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import roc_curve
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn import svm
np.random.seed = 42
```

In [2]:

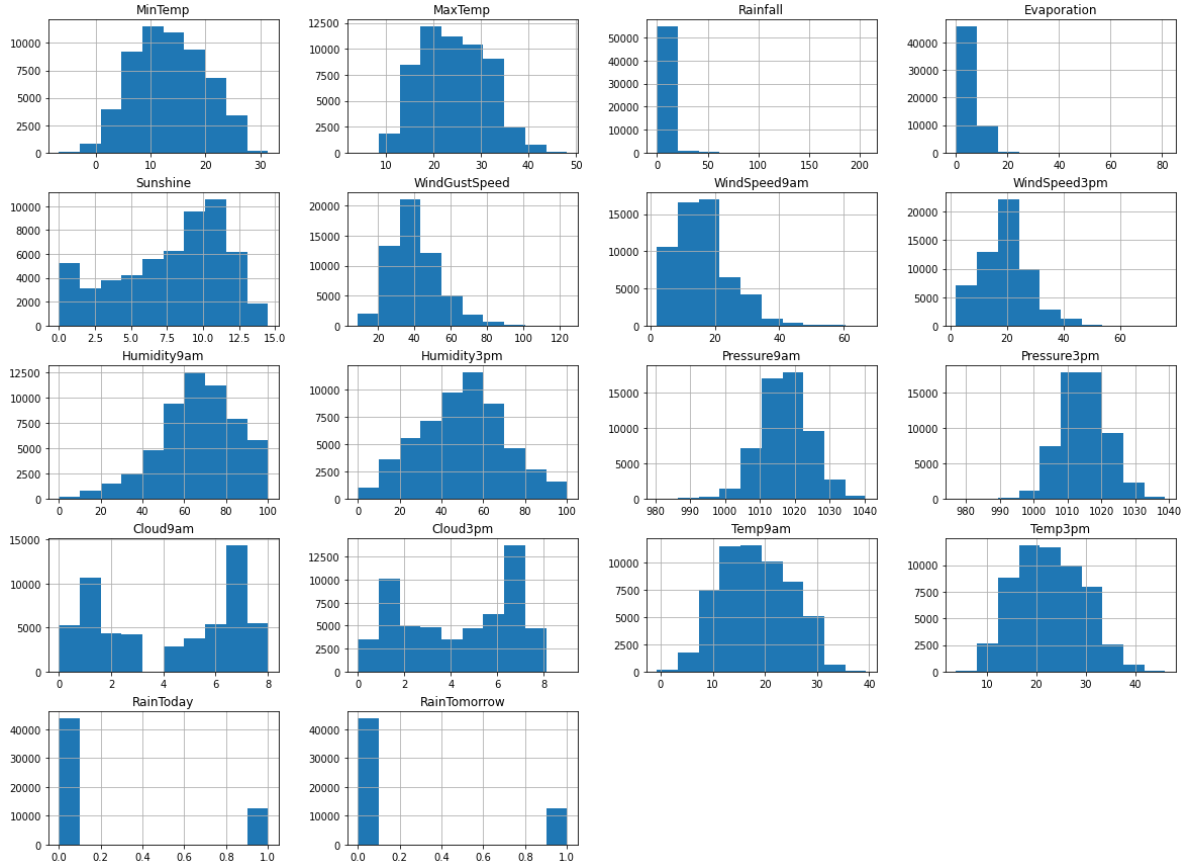
```
df = pd.read_csv("https://raw.githubusercontent.com/mini-pw/2021L-WUM/main/Prace_domowe/Pra
df.head()
```

Out[2]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	Wind
0	17.9	35.2	0.0	12.0	12.3	48.0	6.0	
1	18.4	28.9	0.0	14.8	13.0	37.0	19.0	
2	19.4	37.6	0.0	10.8	10.6	46.0	30.0	
3	21.9	38.4	0.0	11.4	12.2	31.0	6.0	
4	24.2	41.0	0.0	11.2	8.4	35.0	17.0	

In [6]:

```
df.hist(figsize=(20,15))
plt.show()
```



In [7]:

```
X = df.drop("RainTomorrow", axis=1)
y = df[["RainTomorrow"]]
```

In [8]:

```
y.describe()
```

Out[8]:

	RainTomorrow
count	56420.000000
mean	0.220259
std	0.414425
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

Jak można zauważyć mamy znacznie więcej dni bez deszczu niż tych deszczowych. By zachować odpowiednią proporcję tych dni w zbiorze testowym i treningowym, przy podziale użyjemy stratify.

In [23]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0.2)
kfolds = 3
split = KFold(n_splits=kfolds, shuffle=True, random_state=42)
indicators = ["recall", "accuracy", "roc_auc", "f1"]
models_summary = pd.DataFrame(columns = indicators)
```

Modele

Random forest

In [20]:

```
rfc = RandomForestClassifier(n_estimators=400, max_depth=40, max_features=4, random_state=4
rf_model = Pipeline(steps=[('standardscaler', StandardScaler()), ('rf', rfc)])

rf_cv_results = cross_validate(rf_model, X_train, y_train, cv=split, scoring=indicators, n_
```

In [24]:

```
models_summary.loc["RF"]=[np.mean(rf_cv_results["test_recall"]), np.mean(rf_cv_results["tes
                          np.mean(rf_cv_results["test_roc_auc"]), np.mean(rf_cv_results["te
```

In [25]:

```
models_summary.loc["RF"]
```

Out[25]:

```
recall      0.534927
accuracy    0.860090
roc_auc     0.894868
f1          0.627432
Name: RF, dtype: float64
```

SVC

In [40]:

```
svc = svm.SVC(kernel="linear", random_state=42)
svc_model = Pipeline(steps=[('standardscaler', StandardScaler()), ('svc', svc)])

sv_cv_results = cross_validate(svc_model, X_train, y_train, cv=split, scoring=indicators, n
```

In [41]:

```
models_summary.loc["SVC"]=[np.mean(sv_cv_results["test_recall"]), np.mean(sv_cv_results["te
                           np.mean(sv_cv_results["test_roc_auc"]), np.mean(sv_cv_results["t
```

In [42]:

```
models_summary.loc["SVC"]
```

Out[42]:

```
recall      0.505391
accuracy    0.853598
roc_auc     0.882879
f1          0.603289
Name: SVC, dtype: float64
```

Logistic regression

In [44]:

```
knn = KNeighborsClassifier(n_neighbors=5, weights='distance', p=2, n_jobs=-1)
knn_model = Pipeline(steps=[('standardscaler', StandardScaler()), ('knn', knn)])

knn_cv_results = cross_validate(knn_model, X_train, y_train, cv=split, scoring=indicators,
```

In [45]:

```
models_summary.loc["KNN"]=[np.mean(knn_cv_results["test_recall"]), np.mean(knn_cv_results["
                           np.mean(knn_cv_results["test_roc_auc"]), np.mean(knn_cv_results[
```

In [46]:

```
models_summary.loc["KNN"]
```

Out[46]:

```
recall      0.511580
accuracy    0.838222
roc_auc     0.834904
f1          0.582098
Name: KNN, dtype: float64
```

Porównanie modeli

In [57]:

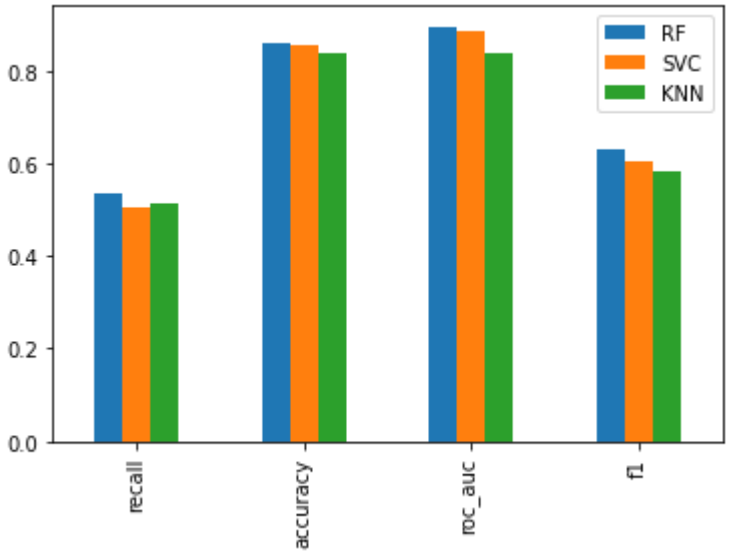
```
models_summary.reset_index()
```

Out[57]:

	index	recall	accuracy	roc_auc	f1
0	RF	0.534927	0.860090	0.894868	0.627432
1	SVC	0.505391	0.853598	0.882879	0.603289
2	KNN	0.511580	0.838222	0.834904	0.582098

In [66]:

```
models_summary.T.plot.bar()
plt.show()
```



Porównując wytrenowane modele możemy zauważyć, że las losowy wypada najlepiej w każdej z obliczonych miar. Oczywiście dobierając nieco inne wartości hiperparametrów wyniki mogłyby być różne. Porównując SVC z KNN możemy zauważyć, że ten drugi ma wyższy wynik recall, jednak wypada znacznie słabiej w pozostałych wartościach. Jeśli jednak chcielibyśmy przewidzieć jak najwięcej deszczowych dni mógłby być on warty rozważenia. Możemy zauważyć, że wszystkie modele mają wysokie accuracy, jednak jeśli weźmiemy pod uwagę, że dni bez deszczu stanowią zdecydowaną większość, wówczas wartości te nie robią żadnego wrażenia. Model który za każdym razem przewidywałby, że nie będzie deszczu, miałby równie wysokie accuracy, jednakże oczywiście byłby bezużyteczny.