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

In [2]:
```python
import matplotlib.pyplot as plt
```

In [3]:
```python
dataframe_train = pd.read_csv('titanic.csv')
dataframe_train.head()
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [4]:
```python
dataframe_train.shape
```

Out[4]:
```
(891, 12)
```

In [5]:
```python
dataframe_train.dtypes
```

Out[5]:
```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

In [6]:
```python
dataframe_train.Survived.value_counts()
```

Out[6]:
```
0    549
1    342
Name: Survived, dtype: int64
```

In [7]:
```python
dataframe_train.Sex.value_counts()
```

Out[7]:
```
male      577
female    314
Name: Sex, dtype: int64
```

In [8]:
```python
dataframe_train.Embarked.value_counts()
```

Out[8]:
```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

In [9]:
```python
dataframe_train.isnull().sum()
```

Out[9]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [10]:
```python
dataframe_train.columns
```

Out[10]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [11]:
```python
# All these columns seems to be irrelevant.
columns_to_drop = ['PassengerId', 'Name', 'Ticket', 'Cabin','Embarked']
```

In [12]:
```python
dataframe_train = dataframe_train.drop(columns_to_drop, axis = 1)
dataframe_train.head()
```

Out[12]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
0	0	3	male	22.0	1	0	7.2500
1	1	1	female	38.0	1	0	71.2833
2	1	3	female	26.0	0	0	7.9250
3	1	1	female	35.0	1	0	53.1000
4	0	3	male	35.0	0	0	8.0500

In [13]:
```python
dataframe_train[dataframe_train['Age'].isna()]
```

Out[13]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
5	0	3	male	NaN	0	0	8.4583
17	1	2	male	NaN	0	0	13.0000
19	1	3	female	NaN	0	0	7.2250
26	0	3	male	NaN	0	0	7.2250
28	1	3	female	NaN	0	0	7.8792
...	...	...	...	...	...	...	...
859	0	3	male	NaN	0	0	7.2292
863	0	3	female	NaN	8	2	69.5500
868	0	3	male	NaN	0	0	9.5000
878	0	3	male	NaN	0	0	7.8958
888	0	3	female	NaN	1	2	23.4500

177 rows × 7 columns

In [14]:
```python
# Smart way of replacing null values in age column by looking into Pclass as reference column
dataframe_train.groupby('Pclass').mean()[['Age']]
```

Out[14]:

	Age
Pclass	
1	38.233441
2	29.877630
3	25.140620

In [15]:
```python
dataframe_train.head()
```

Out[15]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare
0	0	3	male	22.0	1	0	7.2500
1	1	1	female	38.0	1	0	71.2833
2	1	3	female	26.0	0	0	7.9250
3	1	1	female	35.0	1	0	53.1000
4	0	3	male	35.0	0	0	8.0500

In [16]:
```python
def age_approx(cols):
    Age = cols[0]
    Pclass = cols[1]

    if pd.isnull(Age):
        if Pclass == 1 :
            return 39
        elif Pclass == 2:
            return 30
        else:
            return 24
    else:
        return Age
```

In [17]:
```python
dataframe_train['Age'] = dataframe_train[['Age','Pclass']].apply(age_approx, axis = 1)
```

In [18]:
```python
dataframe_train.isna().sum()
```

Out[18]:
```
Survived    0
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
dtype: int64
```

In [19]:
```python
dataframe_train.dtypes
```

Out[19]:
```
Survived      int64
Pclass        int64
Sex          object
Age         float64
SibSp         int64
Parch         int64
Fare        float64
dtype: object
```

In [20]:
```python
# applying one hot encoding on sex col
dataframe_train_one_hot = pd.get_dummies(dataframe_train, columns = ['Sex'])
```

In [21]:
```python
dataframe_train_one_hot
```

Out[21]:

	Survived	Pclass	Age	SibSp	Parch	Fare	Sex_female	Sex_male
0	0	3	22.0	1	0	7.2500	0	1
1	1	1	38.0	1	0	71.2833	1	0
2	1	3	26.0	0	0	7.9250	1	0
3	1	1	35.0	1	0	53.1000	1	0
4	0	3	35.0	0	0	8.0500	0	1
...	...	...	...	...	...	...	...	...
886	0	2	27.0	0	0	13.0000	0	1
887	1	1	19.0	0	0	30.0000	1	0
888	0	3	24.0	1	2	23.4500	1	0
889	1	1	26.0	0	0	30.0000	0	1
890	0	3	32.0	0	0	7.7500	0	1

891 rows × 8 columns

In [22]:
```python
from sklearn.preprocessing import StandardScaler, MinMaxScaler
age_mm_scaler = StandardScaler()
fare_std_scaler = StandardScaler()

dataframe_train_one_hot['Age'] = age_mm_scaler.fit_transform(dataframe_train_one_hot[['Age']])
dataframe_train_one_hot['Fare'] = fare_std_scaler.fit_transform(dataframe_train_one_hot[['Fare']])
```

In [23]:
```python
dataframe_train_one_hot.head()
```

Out[23]:

	Survived	Pclass	Age	SibSp	Parch	Fare	Sex_female	Sex_male
0	0	3	-0.538003	1	0	-0.502445	0	1
1	1	1	0.666578	1	0	0.786845	1	0
2	1	3	-0.236858	0	0	-0.488854	1	0
3	1	1	0.440719	1	0	0.420730	1	0
4	0	3	0.440719	0	0	-0.486337	0	1

In [24]:
```python
dataframe_train_one_hot.columns
```

Out[24]:
```
Index(['Survived', 'Pclass', 'Age', 'SibSp', 'Parch', 'Fare', 'Sex_female',
       'Sex_male'],
      dtype='object')
```

In [25]:
```python
#to find the pairwise correlation
plt.figure(figsize=(20,9));
sns.heatmap(dataframe_train_one_hot.corr(),annot=True,cmap='bwr');
plt.title("Correlation map")
```

Out[25]:
```
Text(0.5, 1.0, 'Correlation map')
```

In [26]:
```python
X = dataframe_train_one_hot[['Pclass', 'Age','SibSp','Parch', 'Fare', 'Sex_female', 'Sex_male']]
y = dataframe_train_one_hot['Survived']
```

In [27]:
```python
X.shape,y.shape
```

Out[27]:
```
((891, 7), (891,))
```

In [28]:
```python
X.head()
```

Out[28]:

	Pclass	Age	SibSp	Parch	Fare	Sex_female	Sex_male
0	3	-0.538003	1	0	-0.502445	0	1
1	1	0.666578	1	0	0.786845	1	0
2	3	-0.236858	0	0	-0.488854	1	0
3	1	0.440719	1	0	0.420730	1	0
4	3	0.440719	0	0	-0.486337	0	1

In [29]:
```python
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,random_state = 6)
```

In [30]:
```python
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

Out[30]:
```
((668, 7), (223, 7), (668,), (223,))
```

In [31]:
```python
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
```

In [32]:
```python
logreg.fit(X_train,y_train)
```

Out[32]:
```
LogisticRegression()
```

In [33]:
```python
# X_test['Fare'] = fare_std_scaler.transform(X_test[['Fare']])
# X_test['Age'] = fare_std_scaler.transform(X_test[['Age']])
```

In [34]:
```python
y_pred = logreg.predict(X_test)
y_pred
```

Out[34]:
```
array([1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
       0, 0, 0], dtype=int64)
```

In [35]:
```python
for i in range(len(X_test)):
              print(logreg.predict_proba(X_test)[i])
```

```
[0.06944503 0.93055497]
[0.91465755 0.08534245]
[0.89714331 0.10285669]
[0.13578224 0.86421776]
[0.55886009 0.44113991]
[0.9378918 0.0621082]
[0.91893442 0.08106558]
[0.88931512 0.11068488]
[0.87205309 0.12794691]
[0.92295005 0.07704995]
[0.64421115 0.35578885]
[0.28386448 0.71613552]
[0.74749826 0.25250174]
[0.88015904 0.11984096]
[0.13234865 0.86765135]
[0.89333168 0.10666832]
[0.87202302 0.12797698]
[0.93910629 0.06089371]
[0.20590868 0.79409132]
[0.4217501 0.5782499]
[0.87205309 0.12794691]
[0.91368276 0.08631724]
[0.09877101 0.90122899]
[0.53799746 0.46200254]
[0.67184381 0.32815619]
[0.90713823 0.09286177]
[0.93623904 0.06376096]
[0.74749826 0.25250174]
[0.78554114 0.21445886]
[0.38204333 0.61795667]
[0.12618751 0.87381249]
[0.87992379 0.12007621]
[0.92911187 0.07088813]
[0.94365134 0.05634866]
[0.44874492 0.55125508]
[0.87221798 0.12778202]
[0.90453319 0.09546681]
[0.79125591 0.20874409]
[0.93469797 0.06530203]
[0.43411643 0.56588357]
[0.57969389 0.42030611]
[0.85337789 0.14662211]
[0.84171173 0.15828827]
[0.90393546 0.09606454]
[0.90802568 0.09197432]
[0.88536701 0.11463299]
[0.87204793 0.12795207]
[0.88447034 0.11552966]
[0.186885 0.813115]
[0.78514978 0.21485022]
[0.88518402 0.11481598]
[0.87205309 0.12794691]
[0.93949836 0.06050164]
[0.87052964 0.12947036]
[0.94823247 0.05176753]
[0.84727946 0.15272054]
[0.41010523 0.58989477]
[0.87219738 0.12780262]
[0.77034373 0.22965627]
[0.49704733 0.50295267]
[0.86735427 0.13264573]
[0.93218741 0.06781259]
[0.89274663 0.10725337]
[0.85244168 0.14755832]
[0.4645479 0.5354521]
[0.22737722 0.77262278]
[0.04440636 0.95559364]
[0.94146896 0.05853104]
[0.95211458 0.04788542]
[0.90984896 0.09015104]
[0.42920373 0.57079627]
[0.73079065 0.26920935]
[0.84325983 0.15674017]
[0.87216045 0.12783955]
[0.89902872 0.10097128]
[0.3462568 0.6537432]
[0.74053187 0.25946813]
[0.35166553 0.64833447]
[0.06734292 0.93265708]
[0.75705937 0.24294063]
[0.88091041 0.11908959]
[0.20901934 0.79098066]
[0.09672421 0.90327579]
[0.87655082 0.12344918]
[0.91742657 0.08257343]
[0.08930221 0.91069779]
[0.90450194 0.09549806]
[0.84184097 0.15815903]
[0.14752368 0.85247632]
[0.82916358 0.17083642]
[0.38626575 0.61373425]
[0.17471269 0.82528731]
[0.87185446 0.12814554]
[0.87140869 0.12859131]
[0.93738512 0.06261488]
[0.87202302 0.12797698]
[0.690303 0.309697]
[0.18698332 0.81301668]
[0.77291527 0.22708473]
[0.43267015 0.56732985]
[0.79129407 0.20870593]
[0.44978169 0.55021831]
[0.87216045 0.12783955]
[0.05998287 0.94001713]
[0.75200637 0.24799363]
[0.86707136 0.13292864]
[0.88933332 0.11066668]
[0.25786211 0.74213789]
[0.19538696 0.80461304]
[0.07786993 0.92213007]
[0.89332727 0.10667273]
[0.86165901 0.13834099]
[0.90777776 0.09222224]
[0.36492866 0.63507134]
[0.86271088 0.13728912]
[0.63465581 0.36534419]
[0.3334547 0.6665453]
[0.89890811 0.10109189]
[0.86863265 0.13136735]
[0.36492331 0.63507669]
[0.38414712 0.61585288]
[0.87202302 0.12797698]
[0.69362811 0.30637189]
[0.87205309 0.12794691]
[0.68931428 0.31068572]
[0.1926451 0.8073549]
[0.5875458 0.4124542]
[0.55121697 0.44878303]
[0.85762059 0.14237941]
[0.57290884 0.42709116]
[0.87199121 0.12800879]
[0.2132274 0.7867726]
[0.07403462 0.92596538]
[0.20776668 0.79223332]
[0.85741075 0.14258925]
[0.56828747 0.43171253]
[0.09400837 0.90599163]
[0.48532344 0.51467656]
[0.8466756 0.1533244]
[0.8524872 0.1475128]
[0.93468668 0.06531332]
[0.79352928 0.20647072]
[0.19525023 0.80474977]
[0.9429992 0.0570008]
[0.80678586 0.19321414]
[0.1997305 0.8002695]
[0.89728322 0.10271678]
[0.35551396 0.64448604]
[0.90092876 0.09907124]
[0.84722067 0.15277933]
[0.1324656 0.8675344]
[0.18048848 0.81951152]
[0.53866725 0.46133275]
[0.36486797 0.63513203]
[0.82004845 0.17995155]
[0.63430646 0.36569354]
[0.72407365 0.27592635]
[0.87216132 0.12783868]
[0.04023148 0.95976852]
[0.92374443 0.07625557]
[0.05643717 0.94356283]
[0.58516987 0.41483013]
[0.95371453 0.04628547]
[0.8491001 0.1508999]
[0.39530486 0.60469514]
[0.65109263 0.34890737]
[0.86262421 0.13737579]
[0.60941464 0.39058536]
[0.87202302 0.12797698]
[0.0859867 0.9140133]
[0.72042384 0.27957616]
[0.6598014 0.3401986]
[0.69841487 0.30158513]
[0.87186479 0.12813521]
[0.36492331 0.63507669]
[0.84325983 0.15674017]
[0.68931428 0.31068572]
[0.31628287 0.68371713]
[0.35775034 0.64224966]
[0.60365914 0.39634086]
[0.13163942 0.86836058]
[0.88094192 0.11905808]
[0.11724544 0.88275456]
[0.78397648 0.21602352]
[0.90452189 0.09547811]
[0.75200637 0.24799363]
[0.186885 0.813115]
[0.36592313 0.63407687]
[0.80816189 0.19183811]
[0.85757732 0.14242268]
[0.80612649 0.19387351]
[0.13249481 0.86750519]
[0.57623364 0.42376636]
[0.70668025 0.29331975]
[0.86254205 0.13745795]
[0.84184712 0.15815288]
[0.93079919 0.06920081]
[0.98457133 0.01542867]
[0.87205309 0.12794691]
[0.87559778 0.12440222]
[0.87667993 0.12332007]
[0.610621 0.389379]
[0.05045812 0.94954188]
[0.74678243 0.25321757]
[0.70668025 0.29331975]
[0.34604763 0.65395237]
[0.94762569 0.05237431]
[0.55949905 0.44050095]
[0.87202302 0.12797698]
[0.25670615 0.74329385]
[0.85594528 0.14405472]
[0.91480172 0.08519828]
[0.33682553 0.66317447]
[0.90460029 0.09539971]
[0.87202302 0.12797698]
[0.23297716 0.76702284]
[0.21250618 0.78749382]
[0.84724956 0.15275044]
[0.86222307 0.13777693]
[0.97362026 0.02637974]
[0.87199121 0.12800879]
[0.58494551 0.41505449]
[0.64722186 0.35277814]
```

In [36]:
```python
from sklearn import metrics
```

In [37]:
```python
metrics.accuracy_score(y_test, y_pred)
```

Out[37]:
```
0.8609865470852018
```

In [38]:
```python
metrics.confusion_matrix(y_test, y_pred)
```

Out[38]:
```
array([[133,   9],
       [ 22,  59]], dtype=int64)
```

In [39]:
```python
print(metrics.classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.86      0.94      0.90       142
           1       0.87      0.73      0.79        81

    accuracy                           0.86       223
   macro avg       0.86      0.83      0.84       223
weighted avg       0.86      0.86      0.86       223
```

In [ ]: