In [6]:
```
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
```

In [7]:
```
df = pd.read_csv('insulin_dosage_prediction.csv') .drop("patient_id", axis=1)
df
```

Out[7]:

	gender	age	family_history	glucose_level	physical_activity	food_intake	previous_m
0	male	68	yes	103.49	9.28	high	
1	female	57	yes	113.35	6.67	high	
2	male	24	yes	127.40	4.14	medium	
3	male	49	yes	138.79	6.80	medium	
4	male	65	no	128.42	4.01	low	
...	...	...	...	...	...	...	
9995	female	33	no	193.35	6.40	high	
9996	male	77	yes	140.01	2.28	low	
9997	female	71	yes	190.90	8.53	medium	
9998	female	33	yes	164.27	7.50	high	
9999	male	52	yes	104.39	5.17	high	

10000 rows × 14 columns

In [33]:
```
print(f"Number of missing values:\n{df.isna().sum()}\nNumber of duplicated rows: {d
```

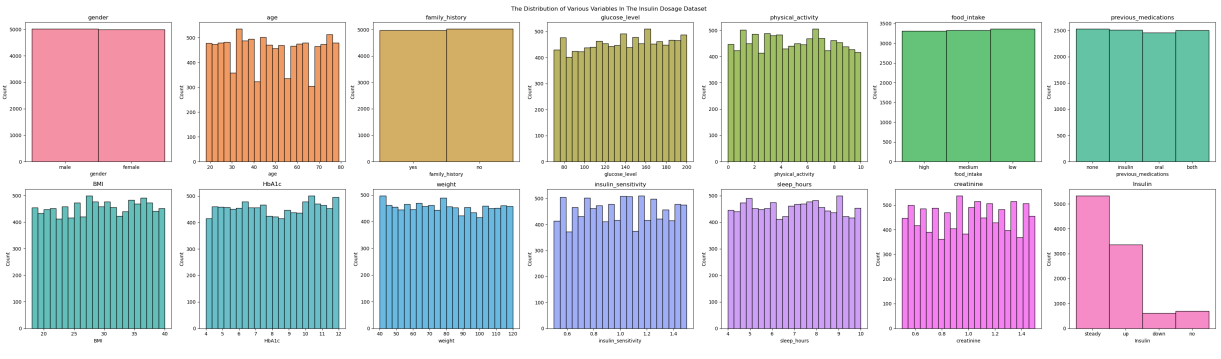
```
Number of missing values:
gender                  0
age                     0
family_history          0
glucose_level           0
physical_activity       0
food_intake             0
previous_medications    0
BMI                     0
HbA1c                   0
weight                  0
insulin_sensitivity     0
sleep_hours             0
creatinine              0
Insulin                 0
dtype: int64
Number of duplicated rows: 0
```

In [34]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   gender                10000 non-null  object 
 1   age                   10000 non-null  int64  
 2   family_history        10000 non-null  object 
 3   glucose_level         10000 non-null  float64
 4   physical_activity     10000 non-null  float64
 5   food_intake           10000 non-null  object 
 6   previous_medications  10000 non-null  object 
 7   BMI                   10000 non-null  float64
 8   HbA1c                 10000 non-null  float64
 9   weight                10000 non-null  float64
 10  insulin_sensitivity   10000 non-null  float64
 11  sleep_hours           10000 non-null  float64
 12  creatinine            10000 non-null  float64
 13  Insulin               10000 non-null  object 
dtypes: float64(8), int64(1), object(5)
memory usage: 1.1+ MB
```

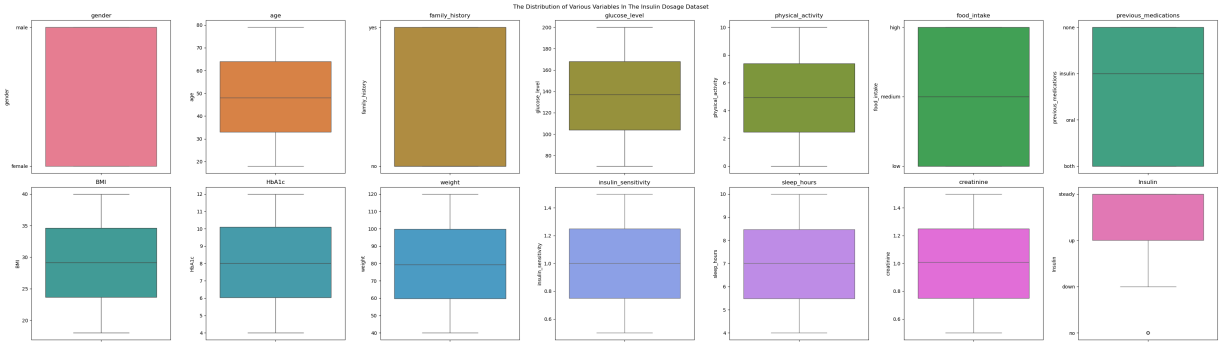
In [35]:
```
# for col in df.columns:
#     print(col)
#     print(np.sort(df[col].unique()))
```

In [36]:
```
fig, ax = plt.subplots(2, 7, figsize=(35,10))

colours = sns.color_palette("husl", 14)

for i, col in enumerate(df.columns):
    sns.histplot(df[col], ax=ax[i//7, i%7], color=colours[i])
    ax[i//7, i%7].set_title(col)

plt.suptitle("The Distribution of Various Variables In The Insulin Dosage Dataset")
plt.tight_layout()
plt.show()
```

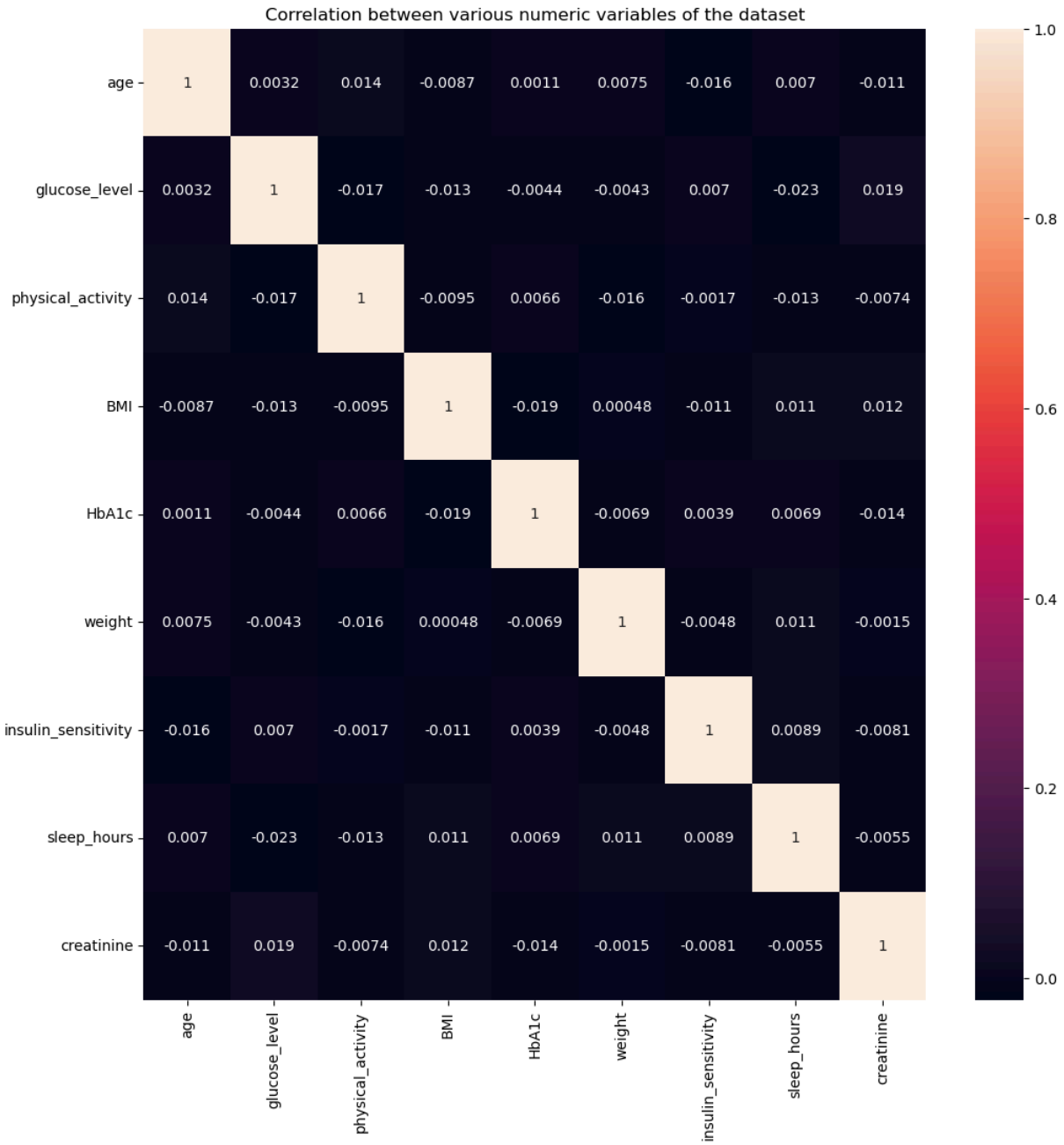


In [74]:
```
fig, ax = plt.subplots(2, 7, figsize=(35,10))

colours = sns.color_palette("husl", 14)

for i, col in enumerate(df.columns):
    sns.boxplot(df[col], ax=ax[i//7, i%7], color=colours[i])
    ax[i//7, i%7].set_title(col)
```

```
plt.suptitle("The Distribution of Various Variables In The Insulin Dosage Dataset")
plt.tight_layout()
plt.show()
```



In [37]:
```
fig, ax = plt.subplots(figsize=(12, 12))
sns.heatmap(df.select_dtypes('number').corr(), annot=True)
plt.title("Correlation between various numeric variables of the dataset")
plt.show()
```



In [10]:
```
from sklearn.preprocessing import StandardScaler, OneHotEncoder, OrdinalEncoder
from sklearn.compose import ColumnTransformer

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
```

In [11]:
```
model = RandomForestClassifier(n_estimators=200)

num_cols = ['age', 'glucose_level', 'physical_activity', 'BMI', 'HbA1c', 'weight',
nominal_cols = ['gender', 'previous_medications']
ordinal_cols = ['family_history', 'food_intake']

preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), num_cols),
        ('nom', OneHotEncoder(sparse_output=False), nominal_cols),
```

```
        ('ord', OrdinalEncoder(), ordinal_cols)
    ]
)

param_grid = {
    'criterion': ['gini', 'entropy'],
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [1, 2, 3, 5, 7, 10]
}
```

In [12]:
```
x = df.drop('Insulin', axis=1)
y = df['Insulin']

x_transformed = preprocessor.fit_transform(x)

x_train, x_test, y_train, y_test = train_test_split(x_transformed, y, test_size=0.3
```

In [ ]:
```
# grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, n_jobs=-
# grid_search.fit(x_train, y_train)

# just so i don't keep accidentally re-running this tedious bit again and again
```

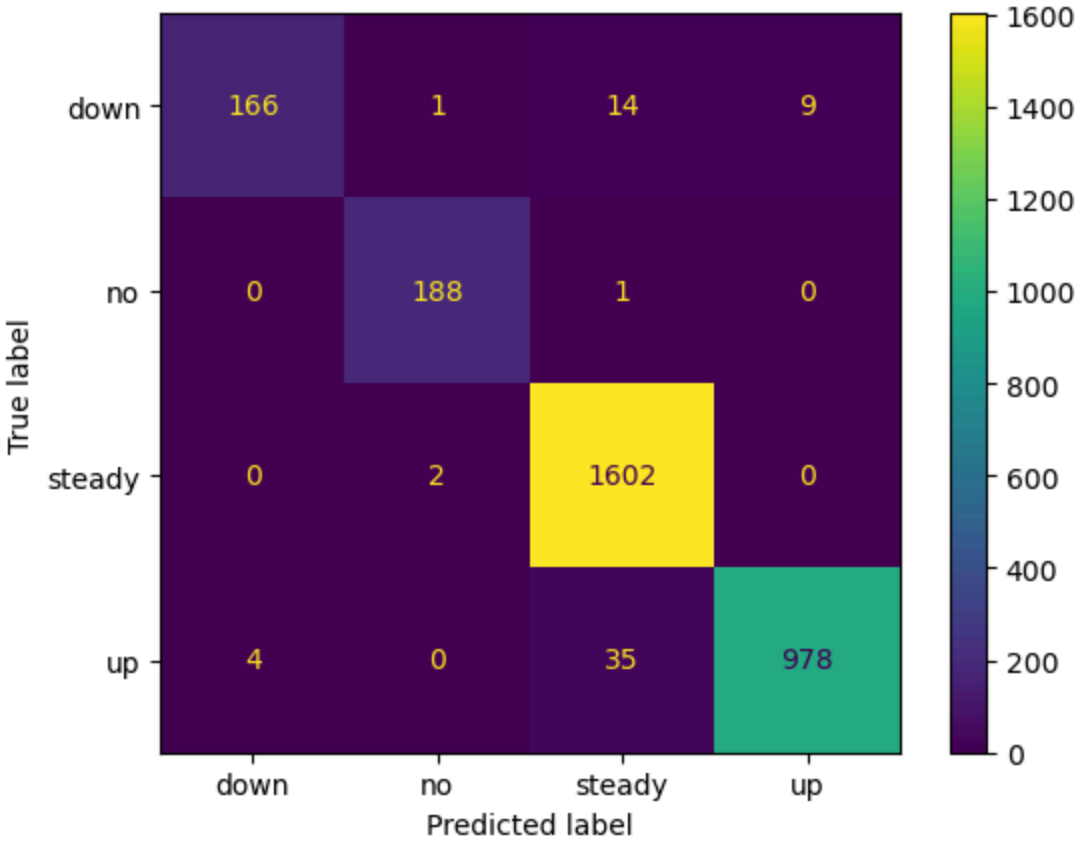
In [42]:
```
from sklearn.metrics import accuracy_score, classification_report, ConfusionMatrixD

best_rf = grid_search.best_estimator_
test_acc = best_rf.score(x_test, y_test)
y_pred = best_rf.predict(x_test)
accuracy_score(y_test, y_pred)
```

Out[42]: 0.978

In [43]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

        down       0.98      0.87      0.92       190
          no       0.98      0.99      0.99       189
      steady       0.97      1.00      0.98      1604
          up       0.99      0.96      0.98      1017

    accuracy                           0.98      3000
   macro avg       0.98      0.96      0.97      3000
weighted avg       0.98      0.98      0.98      3000
```

In [44]:
```
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
plt.grid(False)
plt.show()
```



In [2]:
```
best_rf = model
```

In [40]:
```
from sklearn.utils.validation import check_is_fitted

print(check_is_fitted(best_rf))
```

```
None
```

In [49]:
```
label_binarizer.classes_
```

Out[49]:
```
array(['down', 'no', 'steady', 'up'], dtype='<U6')
```

In [81]:
```
# import matplotlib.pyplot as plt

# from sklearn.metrics import RocCurveDisplay

# from sklearn.preprocessing import LabelBinarizer

# y_score = best_rf.predict_proba(x_test)
# label_binarizer = LabelBinarizer().fit(y_train)
# y_onehot_test = label_binarizer.transform(y_test)

# def graph_roc(var):
#     class_id = np.flatnonzero(label_binarizer.classes_ == var)[0]

#     display = RocCurveDisplay.from_predictions(
#         y_onehot_test[:, class_id],
#         y_score[:, class_id],
#         name=f"{var} VS not {var}",
```

```
#         plot_chance_level=True,
#         despine=True,
#     )
#     _ = display.ax_.set(
#         xlabel="False Positive Rate",
#         ylabel="True Positive Rate",
#         title=f"One-vs-Rest ROC curves:{var} vs not {var}",
#     )

# graph_roc('up')
# graph_roc('down')
# graph_roc('steady')
# graph_roc('no')
```

In [45]:
```
import joblib

joblib.dump(best_rf, 'insulin_dosage_predictor_v2.joblib')
```

Out[45]:
```
['insulin_dosage_predictor_v2.joblib']
```

In [ ]:
```
## all the stuff below this was for debugging and serves no purpose other than show
```

In [1]:
```
import joblib

model = joblib.load("insulin_dosage_predictor_v2.joblib")
```

In [47]:
```
transformed_columns = preprocessor.get_feature_names_out()
transformed_columns
```

Out[47]:
```
array(['num__age', 'num__glucose_level', 'num__physical_activity',
       'num__BMI', 'num__HbA1c', 'num__weight',
       'num__insulin_sensitivity', 'num__sleep_hours', 'num__creatinine',
       'nom__gender_female', 'nom__gender_male',
       'nom__previous_medications_both',
       'nom__previous_medications_insulin',
       'nom__previous_medications_none', 'nom__previous_medications_oral',
       'ord__family_history', 'ord__food_intake'], dtype=object)
```

In [48]:
```
df.columns
```

Out[48]:
```
Index(['gender', 'age', 'family_history', 'glucose_level', 'physical_activity',
       'food_intake', 'previous_medications', 'BMI', 'HbA1c', 'weight',
       'insulin_sensitivity', 'sleep_hours', 'creatinine', 'Insulin'],
      dtype='object')
```

In [49]:
```
gender = 'male'
age = 35
family_history = 'no'
glucose_level = 140.00
physical_activity = 2.00
food_intake = 'low'
previous_medications = 'none'
BMI = 30.00
HbA1c = 8.00
weight = 80.00
```

```
insulin_sensitivity = 1.50
sleep_hours = 8.00
creatinine = 1.30

inputs = [[gender, age, family_history, glucose_level, physical_activity, food_inta
```

In [50]:
```
input_df = pd.DataFrame(inputs, columns=df.columns[:-1])
# input_df = pd.DataFrame(inputs, columns=transformed_columns)
input_df
```

Out[50]:

	gender	age	family_history	glucose_level	physical_activity	food_intake	previous_medi
0	male	35	no	140.0	2.0	low	

In [51]:
```
transformed_data = preprocessor.transform(input_df)
transformed_data
```

Out[51]:
```
array([[-0.74454064,  0.10325266, -1.03306914,  0.14213074, -0.01675114,
         0.01360689,  1.73228436,  0.58525623,  1.03409572,  0.        ,
         1.        ,  0.        ,  0.        ,  1.        ,  0.        ,
         0.        ,  1.        ]])
```

In [53]:
```
prediction = model.predict(transformed_data)[0]
prediction
```

Out[53]:
```
'steady'
```